



Pós-Graduação em Ciência da Computação

**“DESIGN DE SOFTWARE EDUCACIONAL
BASEADO NA TEORIA DOS CAMPOS
CONCEITUAIS”**

POR

Maurício da Motta Braga

Dissertação de Mestrado



Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/~posgraduacao

RECIFE, MARÇO/2006



UNIVERSIDADE FEDERAL DE PERNAMBUCO

CENTRO DE INFORMÁTICA

PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Maurício da Motta Braga

**“Design de software educacional baseado na Teoria dos Campos
Conceituais”**

*ESTE TRABALHO FOI APRESENTADO À PÓS-GRADUAÇÃO
EM CIÊNCIA DA COMPUTAÇÃO DO CENTRO DE
INFORMÁTICA DA UNIVERSIDADE FEDERAL DE
PERNAMBUCO COMO REQUISITO PARCIAL PARA
OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIA DA
COMPUTAÇÃO.*

ORIENTADOR: PROF. DR ALEX SANDRO GOMES

RECIFE, MARÇO/2006

Braga, Maurício da Motta
Design de software educacional baseado na
Teoria dos Campos Conceituais / Maurício da Motta
Braga. – Recife : O Autor, 2006.
99 folhas : il., fig., quadros.

Dissertação (mestrado) – Universidade Federal de
Pernambuco. Cln. Ciência da Computação, 2006.

Inclui bibliografia.

1. Ciência da computação – Inteligência artificial.
2. Informática na educação – Teoria dos campos
conceituais. 3. Software educacional – Avaliação de
aprendizagem. I. Título.

004.855.3
006.3

CDU (2.ed.)
CDD (22.ed.)

UFPE
BC2006-209

Resumo

A percepção pela sociedade do potencial da informática como elemento transformador tem impulsionado as escolas a buscar a melhoria do ensino através da inserção deste recurso nas salas de aula. Como consequência, o uso de software educacional nas escolas vem crescendo, sendo este investimento divulgado pelos estabelecimentos de ensino como prova da qualidade da educação oferecida por essas instituições.

Com o aumento do uso da informática na educação, surge a questão de como avaliar a contribuição das ferramentas computacionais no processo de aprendizagem. Tradicionalmente, a construção e avaliação de software educacional tem sido feita a partir de aspectos intrínsecos desse material. Pouco tem sido dito acerca da contribuição efetiva do uso desses materiais na aprendizagem de conceitos específicos.

Esta dissertação tem como objetivo projetar uma interface educativa a partir do uso de um quadro teórico que possibilite a investigação dos significados que emergem durante o uso de um software educacional para ensino de matemática. Para realizar a investigação, um protótipo do software foi construído e testado com professores do ensino fundamental. Os resultados obtidos permitiram avaliar que propriedades de conceitos emergem durante o uso do software e como as técnicas de *scaffolding* implementadas no software se integram ao processo de resolução de problemas com o software.

Os resultados obtidos nesta pesquisa viabilizaram a construção de um software educacional que será integrado ao ambiente virtual Amadeus, para utilização em cursos e capacitação a distância para professores do ensino fundamental.

Palavras-chave: Software educacional, Avaliação de aprendizagem.

Abstract

Society's perception of the potential of informatics as a transforming factor has led schools to try and improve teaching by introducing this resource into classrooms. As a consequence, the use of educational software in schools has been growing, and this investment is advertised by the schools as proof of the quality of the education provided by them.

With the increase of the use of informatics in education, the question arises as of how to assess the contribution made by computational tools to the learning process. Traditionally, coding and assessment of educational software has been done based on intrinsic aspects of this resource. Very little has been said about the effective contribution of the use of this resource in learning of specific concepts.

This dissertation aims to design an educational interface based on a theoretical framework that makes it possible to investigate the meanings that emerge during the use of educational software in teaching mathematics. To carry out this investigation, a prototype was designed and tested by elementary school teachers. The results obtained made it possible to assess which properties of concepts emerge during the use of the software, and how scaffolding techniques implemented in the software integrate with the process of problem resolution with the software.

The results obtained in this research enabled the coding of an educational software program that will be part of the virtual environment Amadeus, for use in courses and distance learning for elementary school teachers.

Keywords: Educational software, learning evaluation.

À minha família.

Agradecimentos

Escrever a seção de agradecimentos é sem dúvida a parte mais difícil de um trabalho científico. Por mais que nos esforcemos, sempre esquecemos de pessoas importantes. Então, já pedindo desculpas por eventuais omissões, começo este texto.

A Deus, sem o qual nada disso seria possível.

Ao meu orientador, professor Alex Sandro Gomes, cujos ensinamentos foram muito além do tema da pesquisa realizada neste trabalho. A ele agradeço a oportunidade que me deu, bem como a paciência que teve comigo durante a realização desse trabalho.

Ao meu pai, Jacinto Braga, pelo apoio e pelo exemplo de honestidade, dedicação e responsabilidade, valores que carregarei comigo até o fim de meus dias. A minha mãe, Consuelo Braga, a quem devo mais do que poderia descrever nesta seção.

Aos meus irmãos, Ulisses e Virgínia, bem como aos meus cunhados Flávia e Rilke, por estarem presentes em todos os momentos em que precisei deles.

Ao meu avô, Bartolomeu Alves da Mota, pelo exemplo de vida e perseverança. A todos os meus familiares, que fizeram e fazem da minha vida algo que vale a pena ser vivido.

A Janaina, pela paciência comigo nos momentos difíceis e pela sua presença imprescindível. A todos os seus familiares, que me receberam tão bem.

A minha colega de mestrado Ana Emília, que trabalhou comigo durante boa parte da realização deste trabalho, tendo projetado o *scaffolding* utilizado nesta pesquisa. Aos colegas do CCTE pelo companheirismo durante esta jornada.

Aos meus amigos Kenji, Petruski, Sérgio, Leopoldo, Marcelo e Carlos Falcão, que me ensinaram o significado da amizade.

Aos professores da UFPE Alina Spinillo, Verônica Gitirana, Jorge Falcão e Síntria Lautert, pelas sugestões que muito contribuíram para o enriquecimento deste trabalho. Aos professores Décio Fonseca e Fernando Fonseca, pelo muito que pude aprender nas disciplinas deste programa de mestrado, bem como pelas boas conversas que tivemos.

Aos professores e funcionários da Faculdade Metropolitana, pela oportunidade de conhecê-los e poder trabalhar com eles.

A todos que, de uma maneira ou de outra, contribuíram para que eu chegasse até aqui, meus sinceros agradecimentos.

Recife, Março de 2006.

Maurício da Motta Braga

Sumário

<u>1</u>	<u>INTRODUÇÃO</u>	<u>14</u>
<u>2</u>	<u>LEARNWARE E DESENVOLVIMENTO DE SOFTWARE EDUCACIONAL.....</u>	<u>17</u>
2.1	Histórico do software educacional.....	17
2.2	Desenvolvimento e avaliação de software educacional.....	19
<u>3</u>	<u>REFERENCIAL TEÓRICO</u>	<u>23</u>
3.1	Vergnaud e a aprendizagem de conceitos	23
3.1.1	O campo conceitual aditivo	26
3.2	O processo colaborativo de aprendizagem através do Scaffolding	32
3.2.1	Tipos de <i>scaffolding</i>	33
3.3	Etapas para o desenvolvimento de software educacional baseado na Teoria dos campos conceituais e no uso de <i>scaffolding</i>	34
<u>4</u>	<u>METODOLOGIA.....</u>	<u>36</u>
4.1	Análise dos competidores.....	36
4.2	Prototipação	36
4.3	Testes de usabilidade	39
4.3.1	Sujeitos	39
4.3.2	Procedimento	40
4.3.3	Tarefas.....	40
4.3.4	Forma de análise qualitativa	43
<u>5</u>	<u>RESULTADOS</u>	<u>45</u>
5.1	Análise de competidores	45
5.1.1	Ahsha Math	45
5.1.2	Paddle Ball	47

5.1.3	Shuffle Board.....	49
5.1.4	Secret of the Lost City.....	50
5.1.5	Math Logic	51
5.1.6	Comentários sobre os competidores	53
5.1.7	Requisitos levantados a partir da análise de competidores	55
5.2	Prototipação	57
5.2.1	Descrição da interface do protótipo	57
5.2.2	Descrição do <i>scaffolding</i> disponibilizado no protótipo.....	59
5.3	Testes de usabilidade	68
5.3.1	Análise dos invariantes mobilizados pelos sujeitos	69
5.3.2	Análise da integração entre as estratégias de scaffolding e a resolução de problemas.....	78
5.3.3	Discussão dos resultados dos testes de usabilidade	86
5.4	Implementação do software.....	88
<u>6</u>	<u>CONCLUSÕES.....</u>	<u>92</u>
6.1	Contribuições	93
6.2	Dificuldades encontradas.....	93
6.2.1	Interdisciplinaridade	93
6.2.2	Professores como usuários	94
6.3	Trabalhos futuros	94
<u>7</u>	<u>REFERÊNCIAS.....</u>	<u>96</u>

Índice de Figuras

<i>Figura 3.1 – Definição de um conceito segundo Vergnaud.</i>	24
<i>Figura 3.2 – Elementos utilizados no diagrama de Vergnaud.</i>	28
<i>Figura 3.3 – Composição de duas quantidades.</i>	29
<i>Figura 3.4 – Transformação aplicada a uma medida.</i>	29
<i>Figura 3.5 – Comparação entre duas medidas.</i>	30
<i>Figura 3.6 – Composição de transformações.</i>	31
<i>Figura 3.7 – transformação aplicada a uma relação.</i>	31
<i>Figura 3.8 – Composição de relações.</i>	32
<i>Figura 4.1 – Tela de um protótipo [SNY03].</i>	38
<i>Figura 4.2 – Componentes de um protótipo [SNY03].</i>	38
<i>Figura 5.1 - O mago Ahsha.</i>	45
<i>Figura 5.2 - Tela principal do Ahsha Math.</i>	46
<i>Figura 5.3 – Tela de criação de testes.</i>	47
<i>Figura 5.4 – Tela de criação de problemas.</i>	47
<i>Figura 5.5 – Tela do jogo Paddle Ball.</i>	48
<i>Figura 5.6 – Problema apresentado no jogo Paddle Ball.</i>	48
<i>Figura 5.7 – Tela do jogo Shuffle Board.</i>	49
<i>Figura 5.8 – Final de uma partida no Shuffle Board.</i>	49
<i>Figura 5.9 – Tela inicial do jogo.</i>	50
<i>Figura 5.10 – Um dos cenários do jogo S. of The Lost City.</i>	51
<i>Figura 5.11 – Problema exibido pelo Math Logic.</i>	52
<i>Figura 5.12 – Substituição do enunciado pela dica.</i>	52
<i>Figura 5.13 – Tela principal do Gerard.</i>	58
<i>Figura 5.14 – Diagrama de composição para o problema 1.</i>	60
<i>Figura 5.15 - Diagrama de composição com texto explicativo.</i>	61
<i>Figura 5.16 – Funcionamento da conexão entre duas representações.</i>	62
<i>Figura 5.17 – Configuração inicial da ajuda para problemas de composição.</i>	63
<i>Figura 5.18 - Ajuda para problemas de composição após arrastar os quadrados.</i>	64
<i>Figura 5.19 – Configuração inicial da ajuda para problemas de transformação.</i>	65
<i>Figura 5.20 - Ajuda para problemas de transformação após fabricar os quadrados.</i>	65

<i>Figura 5.21 - Configuração inicial da ajuda para problemas de comparação.</i>	<i>66</i>
<i>Figura 5.22 - Ajuda para problemas de transformação após a retirada dos quadrados. 67</i>	
<i>Figura 5.23 – Mensagem utilizada no protótipo.</i>	<i>68</i>
<i>Figura 5.24 - Categorias de análise.</i>	<i>69</i>
<i>Figura 5.25 – Mensagem com link para informações adicionais.....</i>	<i>89</i>
<i>Figura 5.26 – Ajuda para a composição modificada.....</i>	<i>90</i>
<i>Figura 5.27 – Software Gerard.....</i>	<i>91</i>

Índice de Quadros

<i>Quadro 4.1 – Comparação entre protótipos de alta e baixa fidelidade [PRE02].</i>	39
<i>Quadro 4.2 – Problema 1.</i>	41
<i>Quadro 4.3 – Invariante chave do problema 1.</i>	41
<i>Quadro 4.4 – Problema 2.</i>	41
<i>Quadro 4.5 – Invariante chave do problema 2.</i>	42
<i>Quadro 4.6 – Problema 3.</i>	42
<i>Quadro 4.7 – Invariante chave do problema 3.</i>	42
<i>Quadro 4.8 – Problema 4.</i>	42
<i>Quadro 4.9 – Invariante chave do problema 4.</i>	43
<i>Quadro 4.10 – Problema 5.</i>	43
<i>Quadro 5.1 – Comparação entre os competidores.</i>	54
<i>Quadro 5.2 – Botões da barra de tarefas.</i>	58
<i>Quadro 5.3 – Problema 1.</i>	62
<i>Quadro 5.4 – Problema 3.</i>	64
<i>Quadro 5.5 – Problema 5.</i>	66
<i>Quadro 5.6 – Sinais utilizados nas transcrições.</i>	69
<i>Quadro 5.7 – Fragmento 1 do teste.</i>	70
<i>Quadro 5.8 – Fragmento 2 do teste.</i>	71
<i>Quadro 5.9 – Fragmento 3 do teste.</i>	71
<i>Quadro 5.10 – Fragmento 4 do teste.</i>	72
<i>Quadro 5.11 – Fragmento 5 do teste.</i>	72
<i>Quadro 5.12 – Fragmento 6 do teste.</i>	73
<i>Quadro 5.13 – Fragmento 7 do teste.</i>	73
<i>Quadro 5.14 – Fragmento 8 do teste.</i>	74
<i>Quadro 5.15 – Fragmento 9 do teste.</i>	74
<i>Quadro 5.16 – Fragmento 10 do teste.</i>	75
<i>Quadro 5.17 – Fragmento 11 do teste.</i>	75
<i>Quadro 5.18 – Fragmento 12 do teste.</i>	76
<i>Quadro 5.19 – Fragmento 13 do teste.</i>	76
<i>Quadro 5.20 – Fragmento 14 do teste.</i>	77

<i>Quadro 5.21 – Fragmento 15 do teste.</i>	77
<i>Quadro 5.22 – Fragmento 16 do teste.</i>	78
<i>Quadro 5.23 – Fragmento 17 do teste.</i>	79
<i>Quadro 5.24 – Fragmento 18 do teste.</i>	79
<i>Quadro 5.25 – Fragmento 19 do teste.</i>	80
<i>Quadro 5.26 – Fragmento 20 do teste.</i>	81
<i>Quadro 5.27 – Fragmento 21 do teste.</i>	82
<i>Quadro 5.28 – Fragmento 22 do teste.</i>	83
<i>Quadro 5.29 – Fragmento 23 do teste.</i>	83
<i>Quadro 5.30 – Fragmento 24 do teste.</i>	84
<i>Quadro 5.31 – Fragmento 25 do teste.</i>	85
<i>Quadro 5.32 – Fragmento 26 do teste.</i>	86
<i>Quadro 5.33 – Problema 1</i>	89

1 Introdução

Com o avanço da tecnologia, temos observado a inserção da informática em todos os setores da sociedade. Os computadores são hoje uma realidade presente na vida das pessoas que, direta ou indiretamente, os utilizam ao realizar suas atividades diárias. Cada vez mais, as pessoas consideram o uso da informática como condição necessária para a redução de custos e melhoria da qualidade de produtos e processos.

A percepção pela sociedade do potencial da informática como elemento transformador tem impulsionado as escolas a buscar a melhoria do ensino através da inserção deste recurso nas salas de aula. Como consequência, o número de computadores nas escolas vem crescendo, sendo este investimento divulgado pelos estabelecimentos de ensino como prova da qualidade da educação oferecida por essas instituições.

O investimento na aquisição de computadores e software por parte dos estabelecimentos de ensino para uso na educação se justifica, uma vez que há evidências dos efeitos positivos da utilização de software educativo na forma de jogos como ferramenta educacional. Eles melhoram a performance dos alunos em matemática e leitura [NUS99]. Os jogos educacionais proporcionam um ambiente no qual o aprendizado é promovido pelas tarefas estimuladas pelo conteúdo dos mesmos, e as habilidades são desenvolvidas como resultado de jogar o jogo [MCF02].

A utilização por parte das escolas de interfaces educacionais como forma de auxiliar o aprendizado levanta a questão sobre a contribuição efetiva desses artefatos na aprendizagem. Esta questão é de interesse sobretudo da escola, que necessita avaliar a qualidade dos produtos educacionais de forma a poder selecionar o material a ser oferecido aos seus alunos.

O desenvolvimento e avaliação de software educacional é alvo de muitas pesquisas no ambiente acadêmico. As avaliações e técnicas de desenvolvimento, entretanto, costumam direcionar o seu foco para o artefato sendo construído. Pouca atenção é dada à análise da aprendizagem que decorre do uso da ferramenta educacional sendo construída [GOM02].

Os efeitos da pouca atenção dada durante o desenvolvimento à análise da contribuição a aprendizagem oferecida pelo software educacional pode ser visto em boa parte das interfaces educativas disponíveis no mercado. A maior parte dessas interfaces educacionais falha em propor situações educativas significativas [HIN01]. A má qualidade desses produtos tem como resultado o baixo aproveitamento por parte dos alunos, que não encontram situações adequadas para desenvolver as habilidades esperadas com a utilização do software [BRA03].

Partimos do pressuposto que a qualidade de um software educacional deve ser medida a partir da avaliação da aprendizagem que decorre do seu uso. Neste sentido, a pesquisa desenvolvida neste trabalho é motivada pelas seguintes questões: Como avaliar a aprendizagem que emerge a partir do uso de um software educacional? Como deve ser

projetado e avaliado o suporte a aprendizagem dentro desta aplicação? Que etapas devem ser seguidas durante a construção de uma interface educativa de forma a incluir no processo de desenvolvimento a avaliação deste parâmetro?

A avaliação da aprendizagem que emerge do uso de uma interface educacional requer um modelo teórico que informe como se dá a aquisição do conhecimento em um domínio específico. Esta mesma teoria deve também informar como os componentes do domínio escolhido devem ser trabalhados de forma a facilitar o aprendizado do aluno bem como permitir a avaliação do conhecimento adquirido pelo mesmo.

Diante disso, o objetivo geral deste trabalho é projetar um software educacional a partir de um modelo teórico que identifique os requisitos para aprendizagem em um determinado domínio e que estabeleça parâmetros que permitam avaliar, durante a construção, os conhecimentos que emergem durante o uso do mesmo. A definição deste objetivo geral nos levou a estabelecer os seguintes objetivos específicos:

- Identificar etapas que orientem o desenvolvimento de software educacional baseado na teoria dos campos conceituais e no suporte à aprendizagem através de *scaffolding*;
- Realizar uma análise de competidores com objetivo de identificar requisitos para o design de software educacional de forma a atender às necessidades de docentes e aprendizes;
- Desenvolver um protótipo de baixa fidelidade a partir de requisitos oriundos da literatura sobre a aprendizagem de conceitos do campo conceitual das estruturas aditivas e dos resultados obtidos com a análise de competidores; e
- Testar a usabilidade das partes do protótipo de baixa fidelidade relacionadas com a resolução de problemas com usuários representativos para adequar a interface do software às necessidades de aprendizagem dos usuários.

O modelo teórico adotado neste trabalho foi a teoria dos campos conceituais de Vergnaud [VER90] cuja premissa básica é de que o conhecimento está organizado em campos conceituais cujo domínio, por parte do sujeito, ocorre ao longo de um extenso período de tempo, através de experiência, maturação e aprendizagem [VER82]. Este quadro teórico aborda o desenvolvimento cognitivo e a aprendizagem de competências complexas utilizando para isso o próprio conteúdo do conhecimento bem como a análise conceitual do seu domínio.

Este trabalho está organizado da seguinte forma: No capítulo 2 é apresentado um breve histórico do uso da informática na educação e é discutida a questão da avaliação do software educacional. No capítulo 3 é apresentada a teoria dos campos conceituais bem como uma estratégia de suporte a aprendizagem através do *scaffolding*, sendo identificado ao final deste capítulo etapas para construção de software educacional baseado nestes dois paradigmas. No capítulo 4 explicitamos a metodologia utilizada neste trabalho. No capítulo 5 discutimos os resultados obtidos. Por fim, no capítulo 6 apresentamos nossas conclusões e contribuições, comentando também as dificuldades

encontradas durante a realização deste trabalho, encerrando o capítulo mencionando as perspectivas de trabalhos futuros.

2 Learnware e desenvolvimento de software educacional

O reconhecimento de que os usuários de interfaces educacionais possuem características e necessidades que os diferenciam de usuários de aplicativos tradicionais [SOL94] levou à busca por processos de construção de software que guiassem o desenvolvimento de forma a atender as demandas específicas desses usuários. Em resposta a essa demanda, metodologias para concepção e avaliação de software educacional têm sido objeto de estudo por parte de pesquisadores das áreas de educação e computação.

Neste capítulo começaremos com uma definição e um breve histórico das interfaces educativas. Em seguida discutimos como as questões relacionadas à construção e avaliação de software educacional tem sido tratadas em pesquisas que focam o uso da informática na educação.

2.1 Histórico do software educacional

Neste trabalho abordamos a concepção e avaliação de software educacional. Entendemos como software educacional ou *Learnware* todos os programas que têm como meta atuar como mediador entre o aprendiz e o conhecimento, facilitando o aprendizado dos conceitos veiculados por esse artefato. Uma definição mais completa para as interfaces educativas é dada por Silva [SIL98], que afirma que “softwares educacionais são programas de computador que possuem uma proposta de ensino, com um objetivo educacional pré-definido e que se proponha a auxiliar na aprendizagem de conteúdos e habilidades, mediante a utilização de uma interface computadorizada”.

A forma de construir software educacional tem sido modificada ao longo do tempo. Inicialmente, os programas com fins educacionais eram criados de forma a refletir as práticas pedagógicas utilizadas em sala de aula. Neste paradigma, o software é concebido e utilizado como substituto do livro didático. De acordo com Valente [VAL93], este caminho reflete o processo normal de introdução de uma tecnologia na sociedade. Inicialmente procura-se reproduzir o que é realizado em sala de aula para depois gradativamente aperfeiçoar o produto, de forma a explorar todo o seu potencial.

De acordo com Valente [*Ibid*], o uso da informática na educação iniciou-se com o conceito de instrução programada elaborado por B. F. Skinner no início da década de 50. A instrução programada se baseia na divisão do conteúdo a ser ensinado em módulos sequenciais. Ao fim de cada módulo o aprendiz é submetido a uma avaliação do aprendizado dos conceitos vistos. A passagem para o próximo módulo requer a aprovação do aluno no teste realizado ao fim do módulo anterior.

A instrução programada ganhou impulso com o advento do computador, quando passou a ser conhecida pelo nome instrução auxiliada por computador (*Computer-Aided Instruction* ou CAI). A década de 60 assistiu ao crescimento desta modalidade de software educacional, financiada por empresas como IBM, RCA e Digital e pelo governo americano. O alto custo dos computadores, entretanto, limitou o uso desses programas a

universidades, que criavam cursos para serem utilizados por alunos do ensino fundamental.

De acordo com Carraher [CAR90], o software baseado em CAI reflete uma visão empirista do conhecimento, na qual o conhecimento é visto como um conteúdo existente na natureza que pode ser transmitido ao aluno, que assume neste contexto um comportamento passivo, apenas recebendo as informações.

O advento da computação pessoal na forma de microcomputadores nos anos 80 permitiu a utilização dos CAI nas escolas. Desenvolvidos na forma de tutoriais, aplicativos, simuladores, programas do tipo exercício e prática e jogos educacionais, essas interfaces passaram a ser desenvolvidas em larga escala. Ainda hoje são desenvolvidos e utilizados em instituições de ensino como ferramenta de apoio ao processo de aprendizagem.

Os anos 60 também assistiram ao desenvolvimento de software educacional baseado em um paradigma de aprendizagem diferente do utilizado nos CAI. Este modelo utilizava a idéia de micromundos, ambientes criados no computador nos quais o indivíduo pode interagir com objetos e explorar o conhecimento, sendo a linguagem LOGO o mais conhecido dos softwares baseados neste paradigma.

De acordo com Burd [BUR99], a linguagem LOGO foi desenvolvida por Seymour Papert no Massachusetts Institute of Technology (MIT), tendo surgido como resultado da interseção entre os trabalhos de Piaget relacionados ao processo de aquisição do conhecimento com a pesquisa desenvolvida no MIT no campo da inteligência artificial.

Para Papert, o aprendizado requer o engajamento do sujeito na construção do seu próprio conhecimento. De acordo com este autor, o indivíduo aprende quando constrói alguma coisa de seu interesse, caminhando no seu próprio ritmo. O ambiente LOGO propicia isso ao usuário através de uma linguagem de programação de fácil compreensão que permite que o indivíduo interaja com uma tartaruga presente na tela do computador. A tartaruga entende um conjunto básico de comandos que são relacionados a conceitos da geometria. Ao dar ordens à tartaruga, o usuário pode movimentá-la na tela, criando desenhos. A interação do usuário com a tartaruga a partir do uso dos comandos presentes na linguagem permite ao sujeito aprender os conceitos básicos relacionados à geometria espacial de forma prática e lúdica [BUR99].

Da mesma forma que havia ocorrido com o CAI, o uso do LOGO cresceu com o advento do microcomputador, ocorrido nos anos 80. Embora tenha caído em desuso atualmente, o LOGO forneceu uma grande contribuição para a melhoria da qualidade do software educacional ao atrair a atenção para a necessidade de interfaces educacionais que coloquem o aprendiz no centro do processo, tornando-o ativo no processo de aprendizado.

2.2 Desenvolvimento e avaliação de software educacional

De maneira similar às aplicações tradicionais da computação, a demanda por soluções educacionais utilizando a informática levou a uma procura por metodologias que facilitem o processo de construir e avaliar software educacional. Ao destacar aspectos considerados por elas relevantes sobre o design de interfaces educacionais e os seus usuários, essas metodologias funcionam como um guia para equipes de desenvolvimento, que costumam seguir à risca os passos e recomendações contidos nas mesmas.

De forma a se adequar às restrições de tempo e recursos presentes no desenvolvimento de produtos para o mercado, cada metodologia prioriza os aspectos do problema que julga mais importantes para obtenção de uma solução com a máxima qualidade possível em detrimento de outros, considerados de menor importância ou irrelevantes. A necessidade de adequar o projeto a prazos e orçamentos pré-estabelecidos costuma relegar a segundo plano aspectos importantes como o processo de avaliação da interface educativa durante o seu desenvolvimento, prejudicando a qualidade do produto final.

A utilização de um modelo de avaliação para uma classe de software levanta a questão dos objetivos a serem atingidos com esta avaliação. De acordo com Vieira [VIE99] avaliar software educacional implica em “analisar como um software pode ter uso educacional, como ele pode ajudar o aprendiz a construir seu conhecimento e a modificar sua compreensão de mundo elevando sua capacidade de participar da realidade que está vivendo”. Se por um lado podemos considerar que existe pouca discordância em relação a uma definição sobre os objetivos de uma avaliação para interfaces educativas como a fornecida por Vieira, há, entretanto, diversas interpretações quanto a como esta avaliação deve ser realizada e que aspectos a mesma deve privilegiar [VIE99, ATA03, GAM98, CAM94, SIL98].

Algumas metodologias embutem a avaliação dentro de um processo que visa também guiar a construção do software educacional. Como exemplos desses processos podemos citar o Design Centrado no Aprendiz (*Learner-Centered Design*) [SOL94] e o desenvolvimento de software baseado na teoria da atividade [BUR99]. Uma outra metodologia de avaliação de software educacional, a MAQSEI [ATA03], tem como meta servir tanto para avaliações somativas (para software já pronto) quanto para avaliações formativas (para software em desenvolvimento).

A Metodologia para Avaliação da Qualidade de Software Educacional Infantil (MAQSEI) tem como objetivo avaliar a qualidade e a adequação de uma dada interface educacional ao ambiente no qual se pretende utilizar este software. Para isto, esta metodologia utiliza métodos da Engenharia de Usabilidade [NIE93] bem como heurísticas técnicas e pedagógicas específicas para software educacional infantil para realizar a avaliação da interface. De acordo com os autores, o objetivo principal da MAQSEI é atender as necessidades das instituições de ensino, fornecendo um método

que lhes permita selecionar programas educativos compatíveis com a sua proposta pedagógica.

Uma outra abordagem para desenvolvimento de software educacional é a do Design Centrado no Aprendiz (*Learner-Centered Design*) [SOL94]. Esta abordagem destaca a importância do engajamento do usuário na atividade educacional como forma de melhorar o aprendizado. Para isso, o projeto da interface deve contemplar o desenvolvimento de atividades que sejam significativas para o aprendiz.

Essa proposta foi desenvolvida a partir das idéias presentes no Design Centrado no Usuário (*User-Centered Design*) [NOR86] e defende o uso do paradigma construtivista de aprendizagem como forma mais adequada de promover o aprendizado durante o uso de uma interface educacional. Além da facilidade de uso destacada pelo design centrado no usuário, o Design Centrado no Aprendiz enfatiza a importância de contemplar no projeto de interfaces educacionais as necessidades específicas dos aprendizes. Essas necessidades podem ser agrupadas em:

- **Crescimento** – O objetivo da educação é promover o crescimento intelectual do usuário. Um software educacional não deve simplesmente acompanhar o usuário durante a realização de exercícios. É necessário identificar o desenvolvimento do aluno na realização das atividades visando ajudá-lo a construir o conhecimento;
- **Diversidade** – Diferenças individuais, culturais e de gênero possuem influência significativa na escolha do ambiente e dos materiais para os estudantes. Para ser efetivo como ferramenta educacional, o software deve levar em consideração esses fatores; e
- **Motivação** – Diferentemente de interfaces construídas para profissionais, o interesse inicial do aluno, bem como o engajamento produzido pelo software no mesmo são fundamentais para a aceitação da ferramenta, colaborando para que a mesma possa atingir os seus objetivos.

O design centrado no aprendiz prega a necessidade de interfaces educativas oferecerem *scaffold* como forma de auxiliar o aluno a construir o seu conhecimento. Como será visto no capítulo seguinte, *scaffold* é o nome dado ao suporte oferecido ao aluno na realização de suas tarefas por um parceiro mais habilitado. Este mecanismo deve estar presente sempre que o aluno precisar, mas à medida que o aprendiz evolui no domínio da ferramenta e do conhecimento, ele deve desaparecer para não atrapalhar a realização das tarefas por parte do indivíduo.

Por fim, uma abordagem baseada na análise da atividade educacional pode ser encontrada em Burd [BUR99]. Nesse trabalho, o autor defende que a criação de aplicações educativas não pode ser realizada “sem se levar em consideração a atividade educacional para a qual se dirige” [*Ibid*, p. 4]. O desenvolvimento de interfaces educativas deve ser iniciado, segundo o autor, a partir do estudo da atividade educacional na qual o mesmo será inserido.

Neste trabalho, o autor propõe uma metodologia de desenvolvimento de software baseada na análise da atividade educacional. Essa metodologia utiliza a teoria da atividade como forma de entender a dinâmica e as necessidades de uma atividade educacional baseada no paradigma construcionista.

A teoria da atividade em seu formato atual foi desenvolvida por Leontiev a partir das idéias de Vygotsky. Seu objetivo é modelar uma atividade humana, tendo como meta entender como a cultura, as ferramentas e as relações sociais influenciam no desenvolvimento do trabalho realizado pelos indivíduos. Em sua proposta, o autor menciona as seguintes justificativas para o uso da teoria da atividade no desenvolvimento de software:

- **Incorporação de fatores humanos** – Ao abordar de forma sistemática pontos como objetivo, motivação e consciência, a teoria da atividade permite levar em consideração as necessidades dos indivíduos que realizam a atividade em questão;
- **Mudança no enfoque** – A teoria da atividade enfatiza o entendimento de todas as interações que o indivíduo realiza com os objetos e pessoas no seu ambiente de trabalho, e não apenas com o computador;
- **Integração interna** – O uso da teoria da atividade pode facilitar a criação de um vocabulário comum, contribuindo para a integração das diversas especialidades que compõe a IHC (Interface Homem-Computador);
- **Integração com outras áreas** – O fato de ter sido influenciada pelo trabalho de filósofos, psicólogos, antropólogos, linguistas, educadores e outros pode facilitar a integração da engenharia de software com outras áreas;
- **Exploração de novos domínios** – A abordagem da atividade como um trabalho realizado por indivíduos que manipulam artefatos em um contexto social facilita o seu uso em novas áreas como as que utilizam o trabalho em equipe e a educação; e
- **Análise de situações de uso** – A teoria da atividade enfatiza que a atividade será melhor entendida se for analisada no ambiente em que ela é realizada, em detrimento de estudos realizados em laboratórios.

As metodologias mencionadas apresentam algumas similaridades. Embora utilizando técnicas diferentes, elas reconhecem a importância de entender as necessidades dos usuários, evitando restringir o foco apenas ao conteúdo a ser trabalhado pelo software. Reconhecem também a existência de necessidades específicas para interfaces educacionais e definem, cada uma a seu modo, critérios de avaliação para verificar a qualidade da interface construída.

Outro ponto comum entre as metodologias é a ausência de uma avaliação centrada na aprendizagem de conceitos. Como adequar o software às necessidades de aprendizagem dos usuários é um ponto que não é tratado nestes trabalhos, que preferem avaliar a aplicação utilizando critérios que avaliam o produto ou a atividade onde o mesmo será inserido, e não o efeito que o seu uso produz nos usuários. Isso está de

acordo com Gomes [GOM02], que afirma que tradicionalmente a avaliação de software educacional é feita a partir da análise de aspectos intrínsecos desses produtos, pouco sendo dito a respeito das contribuições efetivas destes materiais na aprendizagem.

Como mencionado anteriormente, acreditamos que a qualidade de uma interface educacional deve ser medida a partir da análise da aprendizagem que decorre do seu uso. A definição de um modelo de avaliação da aprendizagem que decorre do uso de um software educacional requer, por sua vez, o uso de uma teoria que explique o processo de aquisição do conhecimento. A teoria de aprendizagem que dá suporte ao método de avaliação de aprendizagem utilizado neste trabalho será vista no capítulo seguinte.

3 Referencial teórico

O entendimento de como a aprendizagem ocorre no indivíduo é um importante requisito para o ensino. A compreensão do processo de aquisição de conhecimento em um domínio específico permite ao educador planejar aulas ricas em situações significativas que facilitem a construção do conhecimento por parte do aprendiz. Uma vez que o objetivo de um software educacional é auxiliar o aprendizado do conteúdo sendo veiculado pelo mesmo, este conhecimento também deve ser levado em consideração na concepção das interfaces educativas.

Neste capítulo apresentamos a teoria de aprendizagem que embasa nossa pesquisa, bem como o campo conceitual utilizado neste trabalho. Em seguida, veremos como uma estratégia de ensino baseado no conceito de *scaffolding* pode ser usada para facilitar o aprendizado. No final, algumas considerações serão feitas sobre como aplicar os princípios vistos anteriormente no desenvolvimento de software educacional.

3.1 Vergnaud e a aprendizagem de conceitos

Ao se deparar com a tarefa de ensinar conceitos matemáticos através do uso de software educacional, algumas perguntas surgem: Como os alunos aprendem conceitos matemáticos? Quais as dificuldades apresentadas pelos aprendizes na aprendizagem desses conceitos em particular? Que situações devem ser trabalhadas junto aos estudantes de forma a facilitar a apreensão dos conceitos envolvidos no aprendizado de um conteúdo específico?

Procuraremos responder a essas perguntas utilizando a teoria dos campos conceituais [VER90]. Conforme lembrado por Moreira [MOR02], a teoria dos campos conceituais pode ser resumida como “uma teoria cognitivista neopiagetiana que pretende oferecer um referencial mais frutífero do que o piagetiano ao estudo do desenvolvimento cognitivo e da aprendizagem de competências complexas, particularmente aquelas implicadas nas ciências e na técnica, levando em conta os próprios conteúdos do conhecimento e a análise conceitual de seu domínio”.

Ao desenvolver este quadro teórico, Vergnaud partiu do trabalho de Piaget, utilizando as estruturas gerais do pensamento identificadas por este, redirecionando o foco para a análise do próprio conteúdo do conhecimento, bem como da identificação das situações e conceitualizações necessárias para o aprendizado deste conhecimento. De acordo com Vergnaud, o conhecimento conceitual emerge a partir de situações-problemas, e as dificuldades apresentadas pelos alunos no aprendizado são específicas ao campo conceitual considerado.

Segundo esta teoria, o conhecimento é organizado em campos conceituais. Um campo conceitual é definido como um conjunto de situações cujo domínio requer, por sua vez, o domínio de vários conceitos distintos. Um conceito (Figura 3.1) é definido como uma tríade de conjuntos (S,I,R) [VER90, VER97], onde:

- **S** é um conjunto de situações que dão sentido ao conceito;
- **I** são os invariantes (propriedades, objetos e relações) que são reconhecidos e utilizados pelos sujeitos ao se deparar com essas situações;
e
- **R** é um conjunto de representações simbólicas utilizadas para representar as situações e os invariantes.

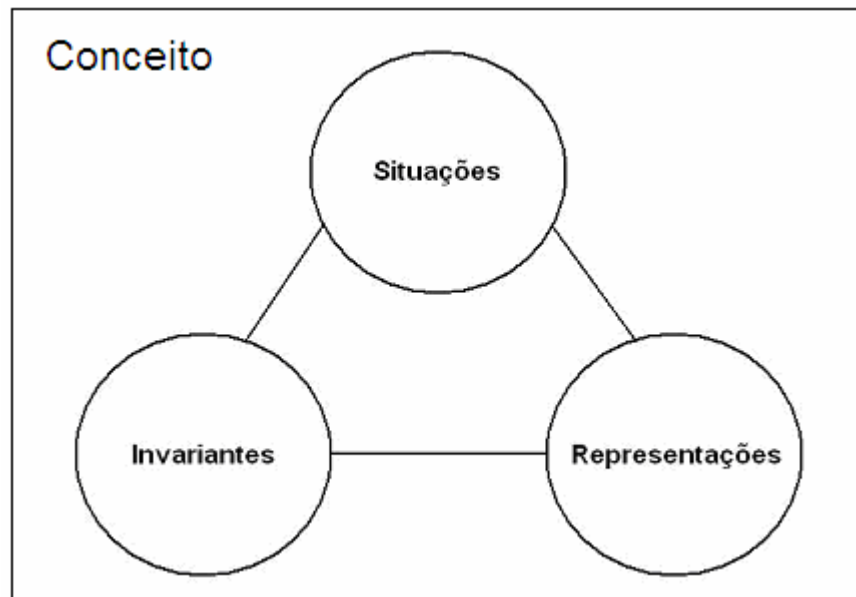


Figura 3.1 – Definição de um conceito segundo Vergnaud.

De acordo com esta teoria, um conceito é progressivamente apreendido por um indivíduo na medida em que este domina mais e mais as situações que o tornam significativo, as propriedades (invariantes) do mesmo e as diferentes formas de representá-lo. Esse aprendizado ocorre ao longo de um extenso período de tempo, durante o qual esses três componentes de um conceito, bem como as relações entre eles, se desenvolvem simultaneamente [GOM00].

Um dos pilares que leva o autor a enfatizar a importância de discutir campos conceituais e não conceitos isolados é a premissa de que a compreensão de um conceito não pode ser obtida a partir de um único tipo de situação, bem como o domínio de uma situação pode envolver mais de um conceito. Essa questão é evidenciada por Magina *et al.* [MAG02], ao explicar o processo de como uma criança aprende a contar. Neste exemplo, as autoras comentam que uma criança aprende a contar juntando os objetos (por exemplo, bolas de gude) para ter certeza que não esquecerá de contabilizar nenhum deles. Se uma criança enfrentar apenas este tipo de situação terá dificuldade em realizar a mesma tarefa quando não puder manipular os objetos. Este exemplo mostra que o conceito de contagem necessita de mais de uma situação para ser definido e entendido pelo aprendiz.

No mesmo exemplo, para contar as bolas de gude, a criança pronuncia a cada bola contada o nome de um número, avançando os números na ordem correta, sem repetir nenhum número, utilizando o princípio de ordenação. Ela também faz uso da correspondência um a um, associando cada número pronunciado a uma bola. Por fim, ao relacionar a última bola a um número, reconhece neste número o cardinal da coleção que ela acabou de contar. Esse exemplo mostra que uma única situação pode envolver mais de um conceito.

Como mencionamos anteriormente, a teoria dos campos conceituais possui raízes nas pesquisas realizadas por Piaget sobre o processo de aquisição de conhecimento. Uma contribuição fundamental do trabalho de Piaget na teoria dos campos conceituais é o conceito de esquema. Este foi originalmente introduzido para explicar o funcionamento das habilidades sensório-motoras bem como da inteligência. Vergnaud admite a existência dos esquemas, mas considera que essas estruturas se referem necessariamente a situações. Os esquemas então são construídos ou modificados pelo indivíduo ao se deparar com uma classe de situações, permitindo ao mesmo lidar com elas.

Segundo Vergnaud [VER97], um esquema é “uma organização invariante do comportamento para uma determinada classe de situações”. As ações realizadas por um sujeito ao se deparar com uma situação são resultado dos esquemas que ele possui para lidar com aquele tipo de situação.

Os esquemas são formados pela combinação dos seguintes elementos [*Ibid*]:

- Objetivos e expectativas;
- Regras que geram ações de acordo com as variáveis da situação, captando informações e realizando validações sobre o resultado das ações;
- Invariantes operatórios (teoremas em ato e conceitos em ato), que permitem ao indivíduo reconhecer os elementos relevantes da situação bem como decidir as regras de ação adequadas à situação; e
- Possibilidade de inferências, que permitem definir, à luz da situação e dos componentes do esquema mencionados anteriormente, as regras e antecipações pertinentes à situação.

Podemos encontrar esquemas em um grande número de atividades que realizamos diariamente. Entre as atividades que realizamos que são governadas por esquemas podemos mencionar: dirigir um automóvel, desenhar e contar o número de elementos em um conjunto de objetos.

Como foi visto, esquema é a organização das ações do sujeito frente a uma classe de situações. Esta definição sugere que a competência de um sujeito em um determinado assunto é função dos esquemas específicos desenvolvidos pelo mesmo para lidar com esse domínio. O ensino de um dado campo conceitual deve então ajudar o aluno a desenvolver um repertório de esquemas que o habilite a lidar com o conjunto de situações relacionadas a esse campo conceitual.

Dentre os elementos que constituem os esquemas, os invariantes operatórios desempenham um papel crucial. São eles os conhecimentos existentes em um esquema, servindo também para diferenciar um esquema de outro. Os invariantes permitem aos indivíduos reconhecer os elementos relevantes em uma dada situação, bem como definir o curso da ação a ser realizada.

Um invariante pode ser explícito ou implícito, dependendo do sujeito ter ou não consciência de sua existência no momento em que faz uso dele. Os invariantes explícitos organizam as ações do usuário, podendo também serem expressos através do uso de representações simbólicas ou palavras. Os invariantes implícitos também estão presentes na ação do sujeito, mas pelo fato de serem conhecimentos tácitos não podem ser compartilhados e discutidos por um aluno em seu ambiente de ensino.

Os invariantes desempenham um papel chave na avaliação do aprendizado. A inferência dos invariantes presentes no raciocínio de um indivíduo pode ser realizada a partir da análise das ações executadas pelo mesmo durante a realização de uma atividade educacional. Essa análise permite avaliar os esquemas que o sujeito desenvolveu para lidar com situações relacionadas a um campo conceitual específico.

Em nossa pesquisa, avaliamos a qualidade do software educacional a partir da análise dos invariantes que a interface leva o usuário a perceber e mobilizar. Uma forma de ajudar o usuário a perceber os invariantes relacionados a um determinado campo conceitual é através do uso de *scaffolding*. Esse assunto será visto na seção 3.2. Na seção seguinte veremos o campo conceitual escolhido para a realização de nossa pesquisa.

3.1.1 O campo conceitual aditivo

A fundamentação para o desenvolvimento da nossa interface educacional encontra-se nas pesquisas sobre o campo conceitual aditivo desenvolvidas pelo pesquisador Gerard Vergnaud [VER91]. Este autor identifica como problemas do tipo aditivo aqueles que envolvem em sua resolução uma adição ou uma subtração. O mesmo ressalta que “existem vários tipos de relações aditivas, e em consequência, vários tipos de (problemas de) adição e subtração” [Ibid, p. 161]. Esta distinção é necessária em virtude do fato de que cada tipo de problema envolve um raciocínio diferente para ser resolvido.

Como lembrado por Franchi [FRA99, p. 159], pesquisas tem mostrado que a principal dificuldade encontrada pelas crianças na resolução de problemas não está em realizar a operação matemática exigida, mas sim nas “operações de pensamento necessárias para estabelecer relações pertinentes entre os dados do problema”. Como a autora nos lembra, “pode haver grande defasagem no domínio pelo aluno de duas situações envolvendo as mesmas operações matemáticas e variáveis diferentes”.

O campo conceitual aditivo, também referenciado como estruturas aditivas, é formado por relações ternárias que podem ser montadas de diferentes maneiras de forma

a oferecer problemas variados. Este campo conceitual é composto de seis categorias elementares de problemas, listados a seguir [VER91]:

- **Categoria 1** – Duas medidas são compostas para dar lugar a uma medida;
- **Categoria 2** – Uma transformação é aplicada em uma medida dando como resultado uma nova medida;
- **Categoria 3** – Uma relação une duas medidas;
- **Categoria 4** – Duas transformações são compostas, dando como resultado uma transformação;
- **Categoria 5** – Uma transformação opera sobre uma relação, dando como resultado uma relação; e
- **Categoria 6** – Duas relações são compostas, dando como resultado uma relação.

Entre os conceitos mobilizados na resolução de situações aditivas, podemos mencionar:

- Conceito de número e sua relação com os conjuntos;
- Conceito de medidas (ex: 5 é maior que 2);
- Conceito de adição;
- Conceito de subtração;
- Conceito de composição de medidas;
- Conceito de transformação de medidas; e
- Conceito de comparação de medidas.

Segundo Vergnaud, o uso de equações matemáticas nas séries iniciais do ensino fundamental traz grandes dificuldades para os aprendizes, que têm dificuldade em relacionar a situação com a operação matemática representada no papel. O autor sugere então o uso de uma representação alternativa, conhecida como a legenda de Vergnaud. Essa representação leva o aluno a reconhecer a categoria do problema que lhe é apresentado, identificando cada componente da situação, o que irá facilitar a interpretação do problema bem como a resolução do mesmo. Os elementos sugeridos por Vergnaud, bem como o significado de cada um deles são vistos a seguir (Figura 3.2).

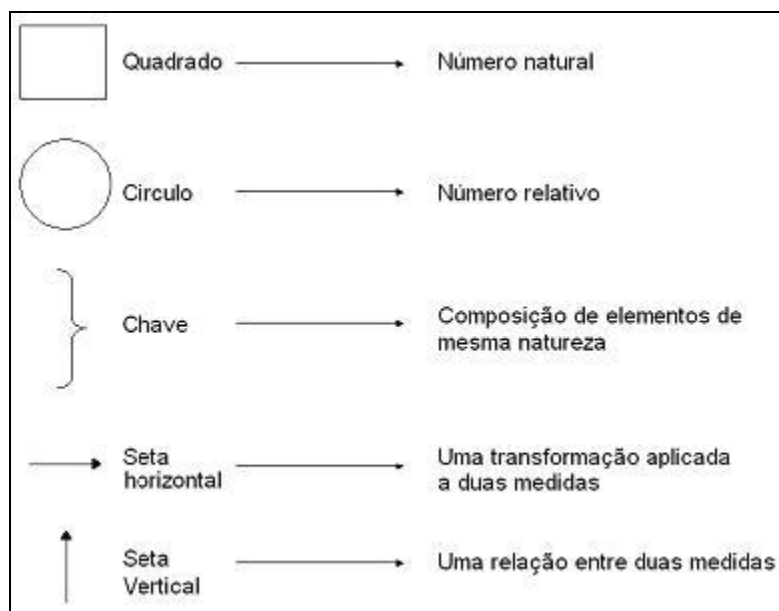


Figura 3.2 – Elementos utilizados no diagrama de Vergnaud.

Em seus trabalhos sobre as estruturas aditivas, Vergnaud identifica diferentes tipos de números que são necessários para a resolução de problemas deste campo conceitual. Os números mais simples são os números naturais, utilizados para indicar uma medida, como o cardinal de uma coleção de objetos. Uma vez que não há sentido em falar em cardinal de uma coleção negativo ou positivo, esses números não possuem sinal.

Além dos números naturais, Vergnaud identifica números relativos, utilizados para representar transformações que são aplicadas aos números naturais bem como relações existentes entre duas medidas. As situações envolvendo perda ou ganho são então representadas com estes números, que podem assumir valores negativos ou positivos.

A seguir veremos em detalhes cada uma das categorias que fazem parte das estruturas aditivas.

3.1.1.1 Primeira categoria

Esta categoria trata dos problemas de composição de quantidades, que formam então um todo. Nesse tipo de situação, pode se juntar duas ou mais partes para obter o todo, ou então subtrair uma ou mais partes do todo para obter uma outra parte. Um exemplo desse tipo de problema, bem como o esquema correspondente a este tipo de situação pode ser visto a seguir (Figura 3.3).

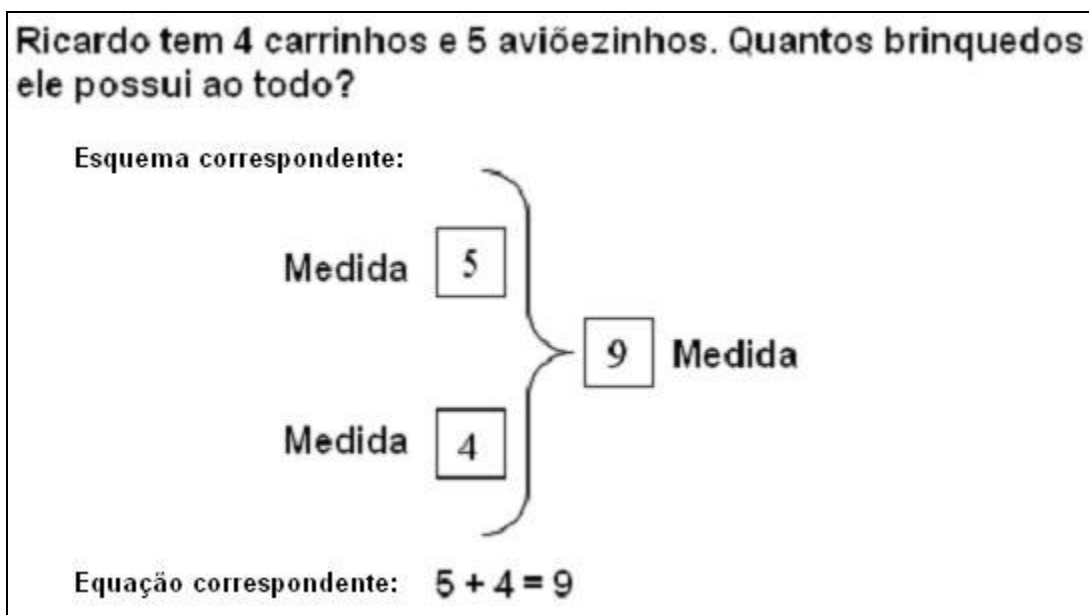


Figura 3.3 – Composição de duas quantidades.

3.1.1.2 Segunda categoria

Esta categoria trata dos problemas em que uma transformação é aplicada sobre uma medida inicial, resultando em uma nova medida. Neste tipo de problema está implícito a idéia de tempo, pois lida-se sempre com um estado inicial (a medida antes da transformação) e um estado final (a medida após ser aplicada a transformação). Neste tipo de situação, pode ser dada a transformação e um dos estados, pedindo-se o outro estado, ou que o aprendiz, a partir de dois estados conhecidos, obtenha a transformação ocorrida. Um exemplo desse tipo de problema, bem como o esquema correspondente a este tipo de situação pode ser visto a seguir (Figura 3.4).

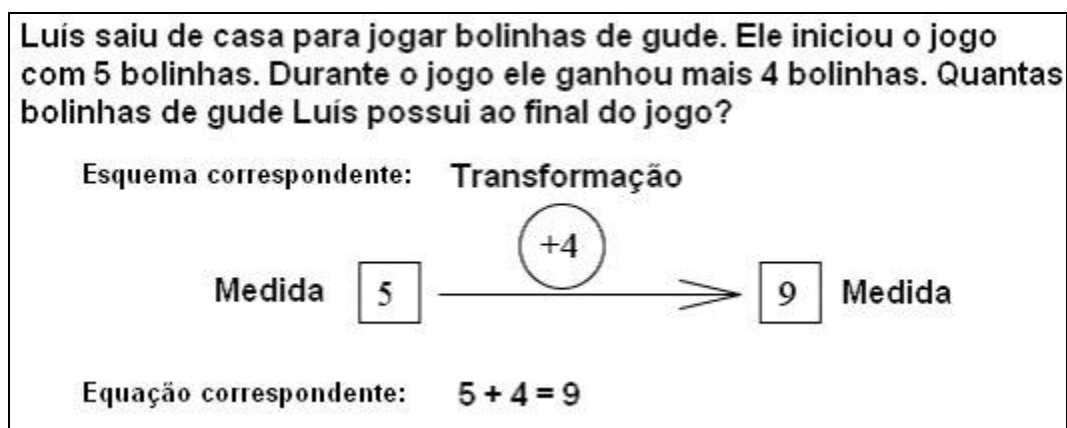


Figura 3.4 – Transformação aplicada a uma medida.

3.1.1.3 Terceira categoria

Esta categoria trata dos problemas em que uma relação existe entre duas medidas. Neste tipo de problema está presente a idéia de comparação, pois lida-se sempre com uma relação que informa a diferença existente entre as duas medidas. Um exemplo desse tipo de problema, bem como o esquema correspondente a este tipo de situação pode ser visto a seguir (Figura 3.5).

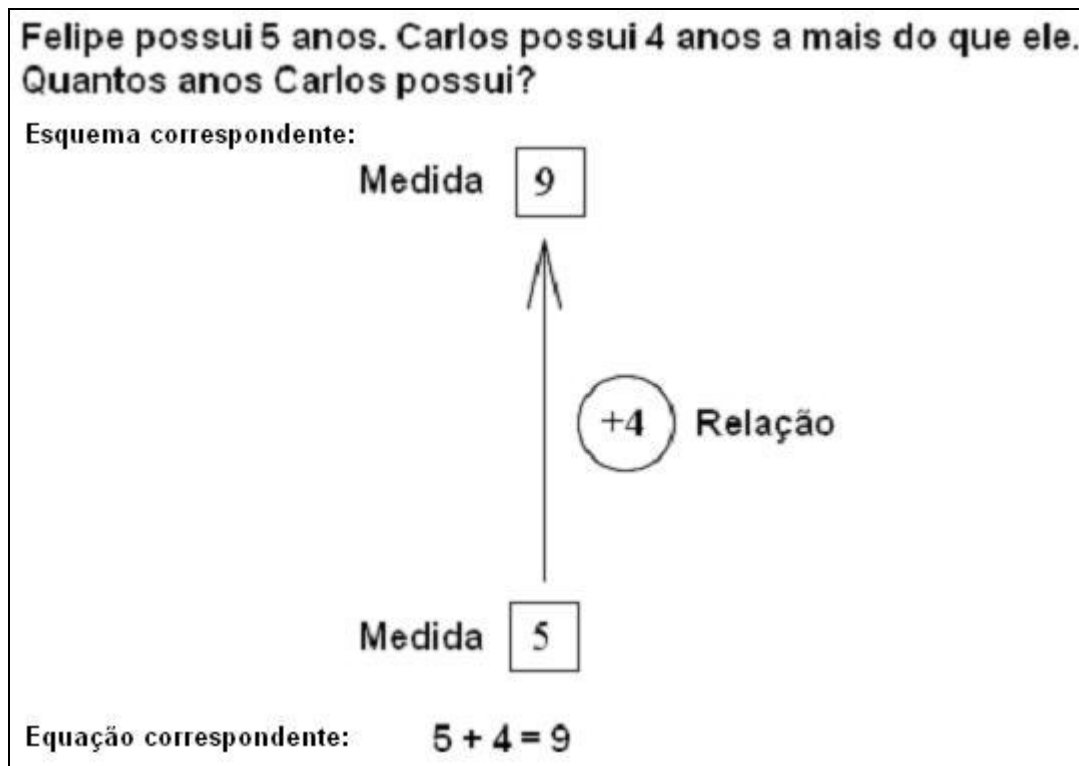


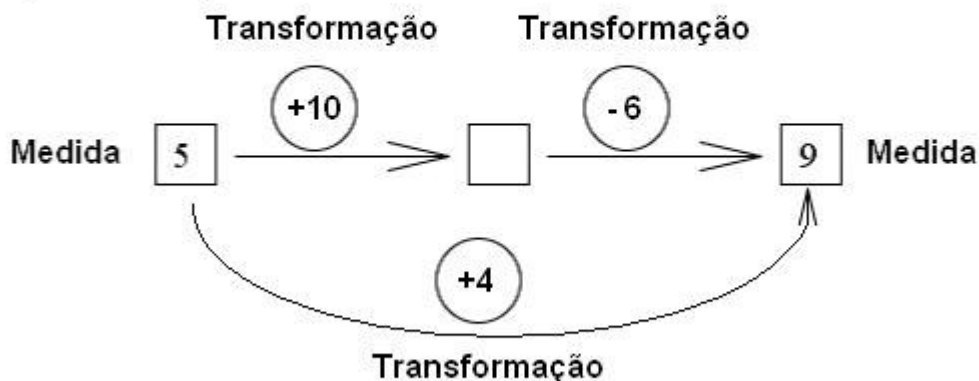
Figura 3.5 – Comparação entre duas medidas.

3.1.1.4 Quarta categoria

Esta categoria trata dos problemas em que uma transformação é aplicada a uma medida inicial, produzindo uma medida em um estado intermediário. Em seguida uma nova transformação é aplicada a essa medida em um estado intermediário, produzindo a medida em seu estado final. Neste tipo de problema encontramos uma composição de transformações, isto é, podemos obter a transformação total aplicada à medida inicial somando as duas transformações. Um exemplo desse tipo de problema, bem como o esquema correspondente a este tipo de situação pode ser visto a seguir (Figura 3.6).

Ricardo saiu para jogar bafo com 5 figurinhas. Ao final da primeira partida ele havia ganho 10 figurinhas. Ao fim da segunda partida ele havia perdido 6 figurinhas. Quantas figurinhas ele ganhou no jogo?

Esquema correspondente:



Equação correspondente: $(+10) + (-6) = (+4)$

Figura 3.6 – Composição de transformações.

3.1.1.5 Quinta categoria

Esta categoria trata dos problemas em que uma transformação é aplicada a um estado relativo, produzindo um novo estado relativo. Um exemplo desse tipo de problema, bem como o esquema correspondente a este tipo de situação pode ser visto a seguir (Figura 3.7).

Felipe tem 15 bolas a mais do que Daniel. Daniel comprou 7 bolinhas. Quantas bolinhas Felipe tem a mais do que Daniel agora?

Esquema correspondente:



Equação correspondente: $(+15) + (-8) = (+7)$

Figura 3.7 – transformação aplicada a uma relação.

3.1.1.6 Sexta categoria

Esta categoria trata dos problemas em que duas relações são compostas, produzindo uma nova relação. Neste tipo de problema encontramos uma composição de relações. Um exemplo desse tipo de problema pode ser visto a seguir (Figura 3.8).

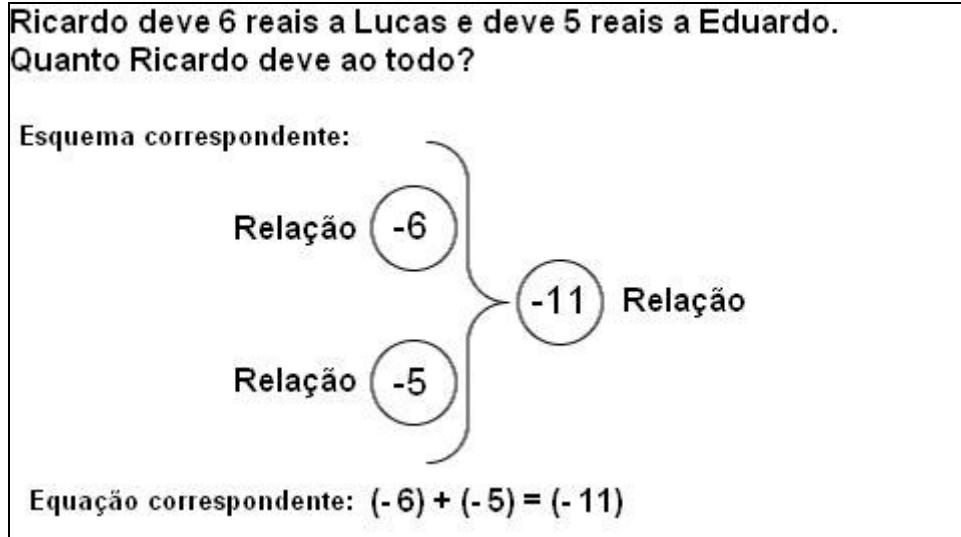


Figura 3.8 – Composição de relações.

Como pode ser visto, as três primeiras categorias apresentavam situações diferentes que exigiam do aluno o mesmo cálculo numérico. Apesar disto, a percentagem de crianças que conseguem resolver a questão definida em cada uma das categorias difere substancialmente, mostrando que a operação matemática não se constitui na principal dificuldade enfrentada pelos alunos ao resolver problemas das estruturas aditivas [MAG02].

A escolha das estruturas aditivas para o software educacional desenvolvido neste trabalho se deve às dificuldades apresentadas pelos alunos do ensino fundamental em lidar com as situações que compõe este campo conceitual [MAG02]. Esta deficiência é causada em parte pelo desconhecimento por parte dos professores dos requisitos para o aprendizado deste assunto. O desenvolvimento desta interface tem então como objetivo a integração do mesmo como um dos micromundos do ambiente virtual de ensino Amadeus, viabilizando a utilização do mesmo em cursos de capacitação a distância voltados para professores do ensino fundamental, através dos quais espera-se despertar os docentes para os requisitos necessários a aprendizagem deste campo conceitual.

3.2 O processo colaborativo de aprendizagem através do Scaffolding

Ao projetar uma interface educacional, novos questionamentos surgem: Como o software pode colaborar com o aprendiz no processo de construção do conhecimento? Que estratégias podem ser utilizadas pela aplicação de forma a levar o aluno a perceber aspectos importantes do domínio sendo trabalhado na interface?

A busca de respostas para essas questões nos levou a conhecer o conceito de *scaffolding*, proposto por Wood, Bruner e Ross [WOO76] de forma a explicar de que forma uma pessoa com maior conhecimento pode ajudar uma pessoa com menor

conhecimento no processo de aprendizagem. A intervenção realizada pelo professor tem como meta permitir ao aluno resolver uma tarefa que está além de sua capacidade. Os autores defendem que esse método de resolução de tarefas assistido por um indivíduo mais capacitado pode facilitar o processo de construção do conhecimento por parte do aluno.

De acordo com estes autores, o *scaffolding* a ser oferecido a um estudante pode ser dividido em 6 categorias. Veremos esse assunto a seguir.

3.2.1 Tipos de *scaffolding*

Existem diversas maneiras de se fornecer *scaffolding* a um estudante, desde fornecer dicas e *feedback* sobre as ações realizadas até realizar uma ou mais etapas da tarefa a título de demonstração. Esses auxílios podem ser classificados de acordo com o objetivo a ser cumprido por cada um no processo de aprendizagem. Como resultado, seis categorias de *scaffolding* foram identificadas [WOO76]:

- **Engajamento** – Consiste em atrair a atenção do aprendiz para a realização da tarefa. Uma forma eficaz de aumentar o envolvimento dos aprendizes em uma tarefa é tornando a mesma atraente, de forma a aumentar a motivação dos alunos;
- **Redução do grau de liberdade** – Tem como objetivo permitir ao aprendiz desenvolver suas habilidades nos componentes da tarefa que ele pode controlar. A redução do grau de liberdade é obtida através da redução dos passos necessários para realizar a tarefa, tornando-a mais simples para o aprendiz. A redução dos passos para a realização da tarefa restringe as ações dos aprendizes às partes da tarefa que ele pode realizar, facilitando o controle das ações e fornecimento de *feedback* por parte do tutor;
- **Manutenção da direção** – Tem como objetivo impedir que o aprendiz se desvie do objetivo ou do caminho necessário para cumprir a tarefa. Durante o processo de realização da atividade, é função do tutor manter a motivação do estudante voltada para aquilo que ele está realizando, para que o mesmo possa aprender os conceitos envolvidos na resolução da tarefa;
- **Aspectos críticos destacados** – Consiste no destaque a ser dado pelo professor, durante o ensino de um conteúdo, às partes que ele julga relevantes para o aprendizado do assunto bem como para a resolução de problemas. O uso dessas partes relevantes por parte do aprendiz pode ser posteriormente utilizado pelo professor como critério para medir o progresso do aluno na aprendizagem do domínio trabalhado;
- **Controle da frustração** – A realização de uma tarefa pode se tornar em uma experiência estressante para o aprendiz quando o mesmo possui dúvidas sobre como resolvê-la. É função do professor procurar diminuir a frustração do aprendiz, aumentando sua motivação para o aprendizado; e

- **Demonstração** – Consiste em desenvolver modelos de soluções para a tarefa ou partes dela, de forma a facilitar a percepção, por parte do aprendiz, dos conceitos e procedimentos relevantes na realização da atividade.

A teoria dos campos conceituais bem como o suporte à aprendizagem através do *scaffolding* forneceram subsídios importantes que permitiram a identificação de passos a serem realizados durante o desenvolvimento de um software educacional baseado nesses dois trabalhos. Esse assunto será visto na próxima seção.

3.3 Etapas para o desenvolvimento de software educacional baseado na Teoria dos campos conceituais e no uso de *scaffolding*

O conhecimento do conjunto de situações e conceitos necessários ao aprendizado de um dado campo conceitual facilita a confecção de interfaces educativas na medida em que traz à tona componentes necessários ao aprendizado desse domínio. O conhecimento desses componentes permite ao desenvolvedor vislumbrar as necessidades relativas ao aprendizado do domínio que deverão ser atendidas pela interface.

A criação de software educacional deve então partir de informações relativas ao campo conceitual tratado pela aplicação. Esta informação pode ser obtida na literatura. Caso o campo conceitual não tenha ainda sido estudado, será necessário a realização de estudos para a definição das situações, conceitos, invariantes e representações necessárias para o aprendizado deste campo conceitual.

Os invariantes identificam as propriedades de um conceito. Conhecer os invariantes relacionados a um conceito viabiliza a apreensão por parte do aprendiz do significado do mesmo. A interface deve então favorecer a percepção dos invariantes relacionados aos conceitos que são necessários para o aprendizado do campo conceitual tratado no software. Esses mesmos invariantes devem ser levados em consideração no momento de avaliar a aprendizagem decorrente da utilização da interface.

Por sua vez, a utilização de *scaffolding* em um software educacional requer a compreensão do papel a ser desempenhado pelo software enquanto parceiro do aluno no processo de aprendizado. Neste sentido, as categorias de *scaffolding* vistas na seção anterior destacam questões importantes a serem atendidas no projeto desse recurso, o qual tem como objetivo final oferecer ajuda adequada às necessidades dos usuários.

Uma forma de adequar o *scaffold* utilizado em um software educacional às necessidades dos aprendizes é investigar o processo de execução da tarefa realizado pelos alunos. A partir da observação dos passos, procedimentos, observações e dificuldades apresentados pelos alunos durante a execução da tarefa é possível formular uma ajuda adequada ao aprendizado do conteúdo em questão. Outras fontes de inspiração a serem consideradas no projeto do *scaffolding* podem ser encontradas no próprio conteúdo a ser veiculado na interface, através da identificação dos conceitos e invariantes necessários a aprendizagem do assunto bem como na análise de programas concorrentes. Uma vez

implementado, o *scaffolding* pode ser avaliado a partir da observação dos usuários utilizando a interface educativa.

A partir do que foi exposto, podemos apontar os seguintes passos para a confecção de uma interface educacional baseada na teoria dos campos conceituais e no uso de *scaffolding*:

- Definir o campo conceitual a ser veiculado na interface;
- Identificar os conceitos necessários ao aprendizado deste campo conceitual;
- Identificar as situações que tornam esses conceitos significativos e oferecer essas situações ao usuário;
- Definir as diferentes representações que serão utilizadas para lidar com essas situações e permitir ao usuário fazer uso delas durante o uso da interface;
- Definir se existirá conexão entre algumas das representações utilizadas e de que forma o usuário poderá utilizar estas conexões para passar informações de uma representação para outra;
- Identificar os invariantes que devem ser capturados pelos usuários ao lidar com essas situações;
- Projetar o *scaffolding* (ajuda para o usuário) a ser utilizado na interface de forma a facilitar a percepção desses invariantes bem como auxiliar o usuário no processo de resolução de problemas com a interface;
- Desenvolver um protótipo para realização de testes com usuários;
- Avaliar os invariantes mobilizados pelo usuário ao utilizar o protótipo bem como a eficácia do *scaffolding* em auxiliar o aprendiz durante o uso da interface educativa; e
- Caso necessário, realizar modificações no protótipo e testar novamente com o usuário. Repetir o ciclo (modificação do protótipo/teste/avaliação) até que o software seja considerado adequado para a aprendizagem dos conceitos veiculados pelo mesmo.

Como pode ser visto, os passos identificados nesta seção não fazem menção a métodos ou técnicas da engenharia de software. O objetivo aqui é apenas identificar o que deve ser realizado durante o desenvolvimento da aplicação educativa. Desta forma, o desenvolvedor tem liberdade para escolher os métodos e ferramentas de desenvolvimento que achar mais conveniente ou com os quais tenha maior familiaridade.

Os passos identificados para a construção de software educacional delineados nesta seção possuem caráter iterativo, exigindo o desenvolvimento rápido de versões da interface educativa que possam ser utilizadas pelos usuários a fim de verificar se os objetivos foram atingidos. Veremos no capítulo seguinte como este requisito pode ser atendido.

4 Metodologia

No capítulo anterior vimos como a teoria dos campos conceituais em conjunto com uma estratégia de ensino baseada em *scaffolding* podem ser utilizadas para identificar etapas a serem seguidas para a construção de software educacional baseado nesses dois paradigmas. Os passos identificados no capítulo anterior, entretanto, não fazem menção a como isso deve ser realizado, fornecendo à equipe de desenvolvimento liberdade para escolher as técnicas e ferramentas de sua preferência.

Neste capítulo descreveremos em detalhes as técnicas que utilizamos para implementar o software educacional desenvolvido neste trabalho.

4.1 Análise dos competidores

A análise de competidores é uma técnica da engenharia da usabilidade [NIE93] que tem como meta estudar produtos concorrentes utilizando critérios específicos de avaliação de forma a conhecer seus pontos fortes e fracos. Entre as vantagens da análise de competidores, podemos citar:

- Avaliar pontos fortes e fracos da concorrência;
- Obtenção de idéias para o design do software;
- Obtenção de requisitos.

De acordo com Maguire [MAG01], avaliar programas concorrentes pode fornecer informações valiosas sobre até que ponto esses produtos atendem os requisitos de seus usuários, sendo também útil para identificar problemas a serem evitados no software em desenvolvimento.

A análise de competidores foi utilizada neste trabalho para elicitar boas práticas de design para interfaces educacionais. Os resultados obtidos permitiram a identificação de requisitos para o software educacional desenvolvido neste trabalho. Os resultados obtidos foram utilizados no processo de definição do protótipo de baixa fidelidade bem como na versão definitiva do software.

4.2 Prototipação

A criação de software em um processo iterativo requer o desenvolvimento rápido de versões utilizáveis do sistema para que as mesmas possam ser avaliadas pelos usuários finais da aplicação. Prototipação é uma técnica de construção de software na qual uma versão inicial de um sistema é disponibilizada rapidamente durante o desenvolvimento com o objetivo de elicitar e validar os requisitos da aplicação [KOT98, p. 73].

Pela necessidade de estar disponível rapidamente para ser avaliado, protótipos costumam ser disponibilizados como versões incompletas, caracterizando-se pela ausência de funcionalidades bem como de confiabilidade. A tecnologia utilizada para

desenvolver o protótipo pode diferir da utilizada no desenvolvimento da versão final [KOT98].

Os protótipos podem ser classificados como descartáveis ou evolucionários. Os protótipos descartáveis tem como objetivo facilitar o entendimento dos requisitos que a equipe ainda não compreende ou não tem certeza de que são necessários para atender às necessidades dos usuários. Desta forma, requisitos já compreendidos não são implementados neste tipo de protótipo. Uma vez que a equipe tenha segurança de que compreendeu todos os requisitos da aplicação, o protótipo é descartado e a versão definitiva do sistema é então construída.

Os protótipos evolucionários possuem um objetivo diferente. Neste caso, a meta é disponibilizar funcionalidade para o cliente o mais rápido possível, para que o mesmo possa ir utilizando o sistema o quanto antes. Neste tipo de protótipo, apenas os requisitos bem compreendidos e que disponibilizam funcionalidades úteis ao cliente são implementados. Os requisitos que a equipe ainda não compreende tem sua implementação retardada para permitir uma análise mais aprofundada dos mesmos.

Os protótipos podem ainda ser classificados de acordo com a fidelidade que apresentam em relação ao produto finalizado. Com vistas a esse aspecto podemos distinguir entre protótipos de alta e baixa fidelidade.

Os protótipos de alta fidelidade têm como característica principal a semelhança com o produto concluído. Estes artefatos costumam utilizar em sua construção materiais que deverão ser utilizados para produzir a versão final do software [PRE02, p. 245]. Ferramentas utilizadas para criação de tais protótipos são os ambientes RAD (acrônimo para *rapid application development* ou desenvolvimento rápido de aplicações) disponíveis comercialmente, como Visual Basic e Delphi.

Os protótipos de baixa fidelidade se caracterizam pela pouca semelhança com o produto concluído. Apesar disto, protótipos de baixa fidelidade mantém a essência da interação com o usuário [NIE93, p. 96], permitindo que os mesmos possam ser usados em testes de usabilidade.

Os protótipos de baixa fidelidade podem ser construídos utilizando materiais simples como papel, caneta e tesoura (Figura 4.1). As telas e demais componentes com os quais os usuários interagem em um software são desenhados a mão ou com o auxílio de um editor gráfico em um computador e recortados para uso no momento adequado (Figura 4.2). A facilidade de construir e modificar um protótipo utilizando papel a fim de verificar hipóteses e descobrir requisitos durante um teste de usabilidade é um dos fatores que levam desenvolvedores a optar por esse tipo de abordagem.

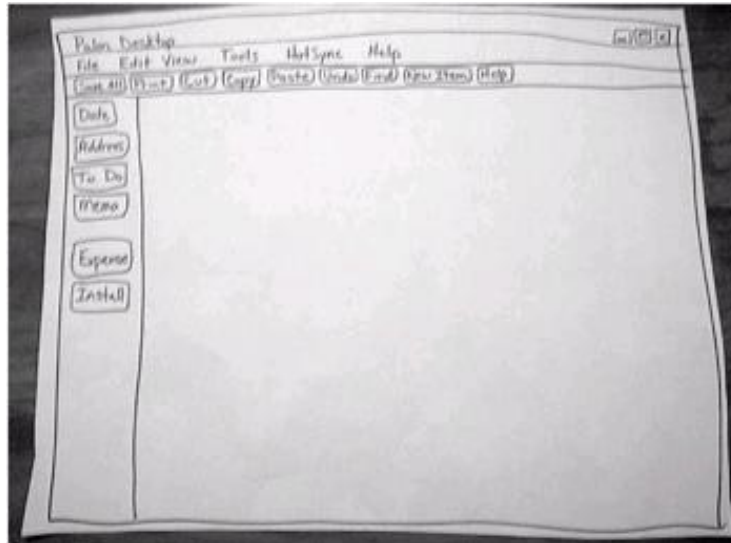


Figura 4.1 – Tela de um protótipo [SNY03].



Figura 4.2 – Componentes de um protótipo [SNY03].

O Quadro 4.1 resume as vantagens e desvantagens da utilização de protótipos de alta e baixa fidelidade:

Quadro 4.1 – Comparação entre protótipos de alta e baixa fidelidade [PRE02].

Tipo de protótipo	Vantagens	Desvantagens
Baixa fidelidade	<ul style="list-style-type: none">• Baixo custo de construção;• Possibilidade de avaliação de múltiplo design;• Útil para comunicar idéias relacionadas ao design; e• Facilidade de construção e modificação.	<ul style="list-style-type: none">• Limitado para avaliação de erros e de questões como performance;• Requer a presença de um facilitador; e• Possibilidades limitadas para navegação.
Alta fidelidade	<ul style="list-style-type: none">• Funcionalidade;• Interatividade; e• Visual do produto final.	<ul style="list-style-type: none">• Custo e tempo de construção;• Modificações podem ser caras; e• Bugs podem prejudicar a avaliação (travamentos).

Neste trabalho, utilizamos um protótipo descartável de baixa fidelidade. O protótipo desenvolvido será visto no capítulo 5.

4.3 Testes de usabilidade

A avaliação da aprendizagem que decorre do uso de uma aplicação educacional requer a realização de testes com usuários representativos. Nesta seção descrevemos como foi realizada a avaliação junto aos usuários.

4.3.1 Sujeitos

Para a realização deste estudo foram convidados 5 professores de matemática com experiência de ensino desta disciplina no ciclo fundamental. Os indivíduos que participaram deste experimento lecionam em instituições de ensino pública e privada da região metropolitana do Recife. Entre os docentes que participaram deste experimento haviam professores pós-graduados, bem como docentes com e sem formação superior.

4.3.2 Procedimento

Tomando como base o desenvolvimento focado na avaliação da aplicação em situações de uso, a pesquisa buscou analisar as ações dos usuários na resolução de problemas aditivos com o protótipo. Desta forma, a análise das ações do usuário com a interface tornou-se o principal elemento de investigação neste trabalho.

O processo de avaliação iniciou-se com uma explanação sobre as estruturas aditivas e o uso da legenda proposta por Vergnaud, bem como sobre a proposta do software em desenvolvimento. Em seguida, os participantes receberam formulários de consentimento descrevendo o experimento de que fariam parte. Os sujeitos foram informados de que era o software e não eles que estavam sendo avaliados, e que poderiam encerrar a sua participação a qualquer momento durante a realização do teste, caso assim o desejassem. Foram alertados ainda que nem sempre iriam receber uma resposta para dúvidas que tivessem em relação aos problemas ou ao funcionamento da interface, pois um dos objetivos era exatamente avaliar se a interface poderia ser utilizada pelo professor sem a ajuda de um usuário mais experiente. Por fim, foram avisados que os testes seriam filmados e que seus nomes não seriam divulgados.

O teste foi realizado com cada usuário individualmente. Cada docente foi filmado resolvendo 5 problemas utilizando o protótipo. Foi solicitado que o professor pronunciasse cada mensagem exibida pelo programa bem como cada ação que pretendia realizar com a interface. Toda a interação do usuário com a aplicação, bem como os comentários do sujeito durante a utilização do protótipo foram registrados também com o auxílio de um gravador.

Após o teste, o vídeo com a gravação do experimento foi exibido para o usuário, e o mesmo foi questionado sobre o motivo de cada ação realizada por ele com o protótipo. Essa parte do experimento teve como objetivo facilitar a identificação dos invariantes mobilizados pelo usuário durante o uso da interface, bem como obter dados qualitativos sobre a usabilidade da interface e a contribuição fornecida pela ajuda disponibilizada na interface no processo de resolução dos problemas com o protótipo. As respostas fornecidas pelos sujeitos nesta etapa foram registradas com o auxílio de um gravador.

4.3.3 Tarefas

As tarefas a serem realizadas pelos sujeitos consistiam nos diferentes tipos de problemas identificados por Vergnaud como pertencentes ao campo conceitual aditivo. Esses problemas envolviam a comparação de duas quantidades, medidas que sofriam transformações originando novas medidas e composição de quantidades. As tarefas resolvidas pelos sujeitos utilizando o protótipo bem como os principais invariantes a serem mobilizados pelos indivíduos durante o uso da interface são descritos a seguir.

Problema 1

O problema 1 tratava de uma composição de duas coleções de objetos distintos. O cardinal de cada coleção era fornecido, cabendo ao sujeito calcular o cardinal do novo conjunto formado pela união dos dois conjuntos. O enunciado deste problema pode ser visto no quadro 4.2.

Quadro 4.2 – Problema 1.

Ao redor da mesa estão sentados 4 garotos e 7 garotas. Quantas pessoas estão sentadas ao redor da mesa?

O invariante chave que deveria ser mobilizado pelo usuário durante o uso da interface educacional, necessário para a resolução do problema 1, consistia em perceber que, se não há interseção entre as duas coleções de elementos, o cardinal da coleção formada com objetos dos dois conjuntos é igual a soma do cardinal de cada coleção considerada individualmente. Este invariante pode ser expresso da seguinte forma (quadro 4.3):

Quadro 4.3 – Invariante chave do problema 1.

$$\text{Card}(A \cup B) = \text{Card}(A) + \text{Card}(B) \quad \text{desde que } A \cap B = \emptyset$$

Neste invariante, representamos por A o conjunto formado pelos garotos e por B o conjunto formado pelas garotas.

Problema 2

O problema 2 tratava de uma transformação aplicada a uma medida. Era informado o estado inicial bem como o estado final da medida, cabendo ao sujeito calcular a transformação ocorrida. O enunciado deste problema pode ser visto no quadro 4.4.

Quadro 4.4 – Problema 2.

Ricardo saiu de casa para jogar bola de gude. Ao sair de casa ele possuía 6 bolas. Após o jogo ele tinha duas bolas. O que aconteceu no jogo?

O invariante chave que deveria ser mobilizado pelo usuário durante o uso da interface educacional, necessário para a resolução do problema 2, consistia em perceber que a transformação aplicada a uma medida pode ser encontrada a partir da diferença entre o estado final da medida e o estado inicial da mesma. Este invariante pode ser expresso da seguinte forma (Quadro 4.5):

Quadro 4.5 – Invariante chave do problema 2.

$$\text{Se } F = I + T \text{ então } T = F - I$$

Neste invariante, I representa o estado inicial da medida, F é o estado final da medida e T é a transformação direta que leva I a F.

Problema 3

O problema 3 tratava de outro problema de transformação aplicada a uma medida. Era informado a transformação bem como o estado final da medida, cabendo ao sujeito calcular o estado inicial. O enunciado deste problema pode ser visto no quadro 4.6.

Quadro 4.6 – Problema 3.

Maria comprou uma caixa de bombons por 4 reais e ainda ficou com 4 reais. Quanto ela possuía antes de fazer a compra?

O invariante chave que deveria ser mobilizado pelo usuário durante o uso da interface educacional, necessário para a resolução do problema 3, consistia em perceber que o estado inicial pode ser obtido aplicando a transformação inversa ao estado final da medida. Este invariante pode ser expresso da seguinte forma (quadro 4.7):

Quadro 4.7 – Invariante chave do problema 3.

$$\text{Se } F = T(I) \text{ então } I = T^{-1}(F)$$

Neste invariante, I representa o estado inicial da medida, F é o estado final da medida, T é a transformação direta que leva I a F e T^{-1} é a transformação inversa que permite obter I a partir de F.

Problema 4

O problema 4 tratava de um problema de comparação, consistindo em duas medidas unidas por uma relação. Era informada a primeira medida (o referente) bem como a relação entre esta e a segunda medida (o referido), cabendo ao sujeito calcular a segunda medida. O enunciado deste problema pode ser visto no quadro 4.8.

Quadro 4.8 – Problema 4.

Ricardo tem 6 anos, Carlos tem 4 anos a mais do que ele. Quantos anos tem Carlos?

O invariante chave que deveria ser mobilizado pelo usuário durante o uso da interface educacional, necessário para a resolução do problema 4, consistia em perceber que a segunda medida pode ser obtida somando a primeira medida com a relação existente entre as duas medidas. Este invariante pode ser expresso da seguinte forma (quadro 4.9):

Quadro 4.9 – Invariante chave do problema 4.

$$M2 = M1 + R$$

Neste invariante, M1 é o referente, M2 é o referido e R é a relação entre as duas medidas, a qual permite obter M2 a partir de M1.

Problema 5

O problema 5 tratava de outro problema de comparação, com a diferença de que desta vez a relação entre as medidas era negativa. Era informada a primeira medida (o referente) bem como a relação entre esta e a segunda medida (o referido), cabendo ao sujeito calcular a segunda medida. O enunciado deste problema pode ser visto no quadro 4.10.

Quadro 4.10 – Problema 5.

Carlos tem 7 reais e Luiz tem 6 reais a menos do que ele. Quantos reais tem Luiz?

O invariante necessário para a resolução do problema 5 é o mesmo do problema 4.

4.3.4 Forma de análise qualitativa

Os dados obtidos durante a realização dos testes foram classificados e analisados com o auxílio do software QSR NUD*IST [QSR06], aplicativo que tem como propósito a análise de dados qualitativos. A pesquisa qualitativa permite ao pesquisador investigar os motivos que levaram o sujeito a realizar uma ação com a interface ou fornecer uma resposta durante a resolução de um problema. Este enfoque é importante quando se deseja conhecer o raciocínio utilizado pelo sujeito durante a realização de uma tarefa.

A análise dos dados dos testes teve como objetivo identificar:

- Os invariantes mobilizados pelos usuários no uso do protótipo;
- A contribuição da ajuda disponibilizada na forma de *scaffolding* ao processo de resolução dos problemas com a interface educacional.

Em um primeiro momento, procuramos na ação dos usuários com a interface e na explicação fornecida pelos mesmos para cada ação a presença de invariantes relacionados

aos problemas resolvidos por esses sujeitos. Em um segundo instante, analisamos a contribuição do sistema de ajuda na forma de *scaffolding* ao processo de resolução de problemas com o software. As informações obtidas foram utilizadas para aperfeiçoar o projeto da aplicação, visando a implementação definitiva do software.

5 Resultados

Este capítulo discute os resultados obtidos a partir das técnicas descritas no capítulo 4. Inicialmente apresentamos os resultados obtidos com a análise de competidores. Em seguida, apresentamos o protótipo desenvolvido para ensino das estruturas aditivas. Por fim, discutiremos os resultados obtidos durante os testes realizados com o protótipo junto aos usuários.

5.1 Análise de competidores

A análise de competidores realizada neste trabalho teve como objetivo a coleta de requisitos relacionados aos recursos disponibilizados nestes programas que consideramos importantes para o desenvolvimento de um software educacional.

Nesta etapa foram analisadas interfaces que tem como objetivo colaborar para o aprendizado do campo conceitual aditivo. A escolha dos programas foi realizada em função do campo conceitual trabalhado nessas interfaces ser o mesmo do estudo de caso desenvolvido neste trabalho. Para a realização desta etapa foram escolhidas cinco interfaces educativas disponíveis no mercado. Os programas escolhidos foram: Ahsha Math [KNI06], Paddle Ball [GRE93], Shuffle Board [GRE93], Secret of The Lost City [DAV06], e Math Logic [CAL06].

5.1.1 Ahsha Math

O Ahsha Math é um tutor interativo que se propõe a auxiliar o aprendizado das estruturas aditivas e multiplicativas. Para isto, o programa administra testes em que uma operação matemática é exibida na tela para o usuário resolver, sob a supervisão de um agente de interface que conduz o usuário durante o uso do software. O agente tem como função instruir o usuário sobre o uso da aplicação bem como fornecer *feedback* sobre as ações realizadas pelo usuário com a interface.

Na primeira vez que o software é executado, o agente Ahsha, um simpático mago, aparece e inicia o diálogo com o usuário. Após se apresentar, o agente solicita que o indivíduo informe o seu nome (Figura 5.1). O agente irá utilizar o nome do usuário quando se dirigir a ele durante o uso do programa.



Figura 5.1 - O mago Ahsha.

Uma vez informado o nome, o usuário é conduzido à tela principal do programa (Figura 5.2). Nesta tela temos as opções disponíveis para configuração do software bem como os controles disponíveis para a resolução dos problemas.

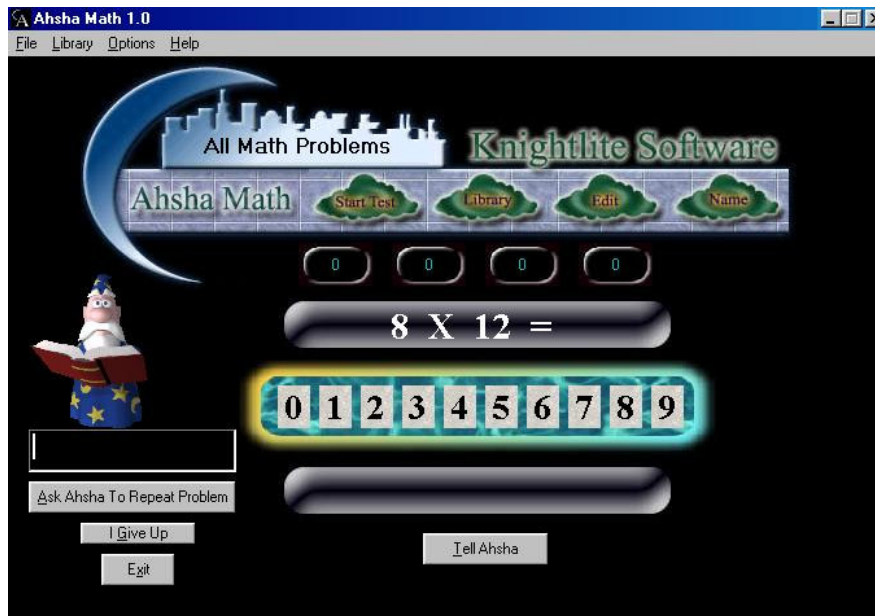


Figura 5.2 - Tela principal do Ahsha Math.

O uso do software se inicia ao selecionar a nuvem com a frase *start test*. A partir daí, um problema será exibido para que o usuário forneça a resposta. O aprendiz pode digitar o resultado na caixa de texto ou selecionar na tela os números que compõe a resposta, pressionando em seguida o botão “Tell Ahsha”. Caso a resposta esteja correta, o agente irá parabenizar o aprendiz e um novo problema será apresentado. Em caso de erro, o agente informará o usuário e rerepresentará o problema. O usuário poderá também desistir do problema clicando no botão “I give up”. Neste caso, a resposta correta será informada e um novo problema será exibido. Esse procedimento irá se repetir até que o usuário complete o teste. Ao final do mesmo, o usuário receberá o resultado.

O Ahsha permite o planejamento de atividades a serem realizadas em sala de aula. Selecionando a opção *Library*, pode-se escolher que tipos de questões (adição, subtração, multiplicação e divisão) serão oferecidas para o usuário resolver (Figura 5.3). Através da opção *Edit*, podemos adicionar novos problemas à base de dados (Figura 5.4). Testes com uma quantidade pré-definida de questões podem ser criados para serem ministrados pelo agente de interface. Esses recursos são úteis na medida em que permitem ao professor criar atividades compatíveis com o planejamento do curso ministrado por ele.



Figura 5.3 – Tela de criação de testes.

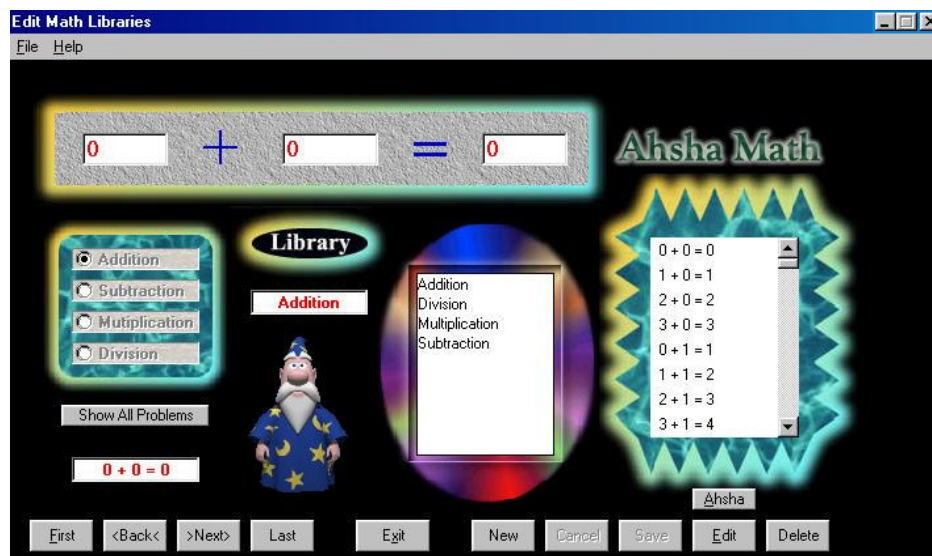


Figura 5.4 – Tela de criação de problemas.

5.1.2 Paddle Ball

O Paddle Ball é um jogo que faz parte de um pacote de programas educacionais chamado Kids Math. O Paddle Ball fornece um ambiente no qual o jogador movimenta uma raquete e deve acertar com uma bola animais que se situam no topo da tela. Quando um objeto é atingido, o mesmo é retirado do jogo. A interface do jogo pode ser vista a seguir (Figura 5.5).



Figura 5.5 – Tela do jogo Paddle Ball.

De tempos em tempos o jogo é interrompido e o usuário é solicitado a resolver um problema de adição ou subtração na forma algébrica (Figura 5.6). O problema apresentado utiliza em sua definição os animais que ainda estão presentes no jogo, permitindo ao usuário contar os animais e oferecer a resposta. Caso o usuário erre duas vezes, a resposta correta é exibida, e uma animação é apresentada mostrando como contar os animais presentes na tela para resolver o problema. Após o usuário informar a resposta, o jogo é reiniciado do ponto onde havia sido interrompido. É possível configurar o programa para mostrar apenas problemas envolvendo adição ou subtração. O *feedback* fornecido pelo software utiliza mensagens, animações e efeitos sonoros, contribuindo para aumentar o interesse pelo mesmo.

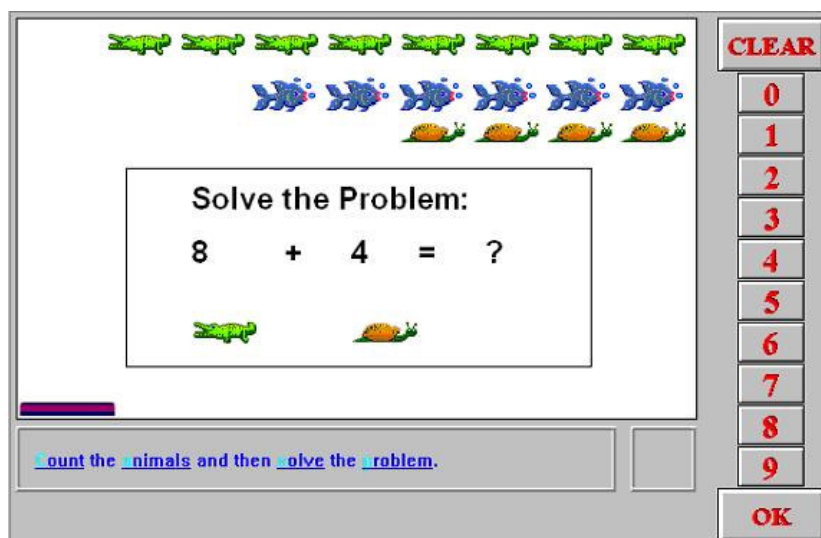


Figura 5.6 – Problema apresentado no jogo Paddle Ball.

5.1.3 Shuffle Board

O Shuffle Board é outro jogo que faz parte do pacote de programas Kids Math. Este jogo fornece um ambiente no qual o jogador lança uma bola em uma mesa que contém algumas seções demarcadas com números. O jogador deve controlar a velocidade inicial da bola de forma a fazer com que ela pare em uma das seções mencionadas anteriormente (Figura 5.7).

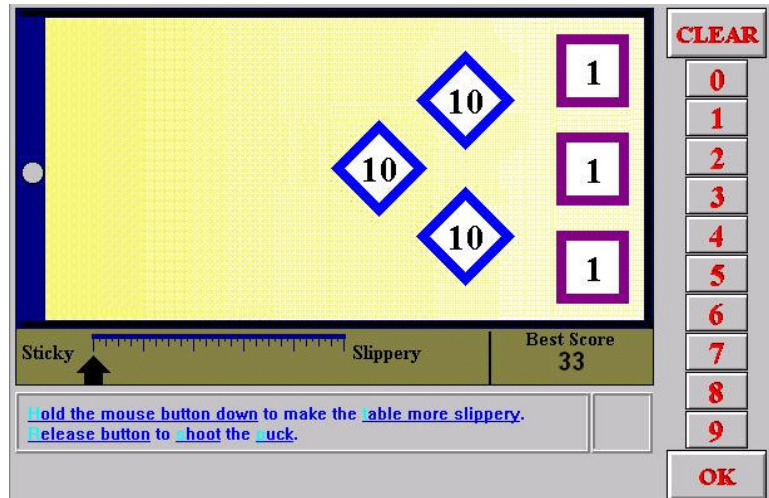


Figura 5.7 – Tela do jogo Shuffle Board.

Ao parar dentro de uma das seções demarcadas com um número, a bola muda sua cor para vermelho, de forma a facilitar a verificação de que a bola está dentro da seção. Após o arremesso de cinco bolas, o problema pergunta quantos pontos foram obtidos pelo usuário (Figura 5.8).

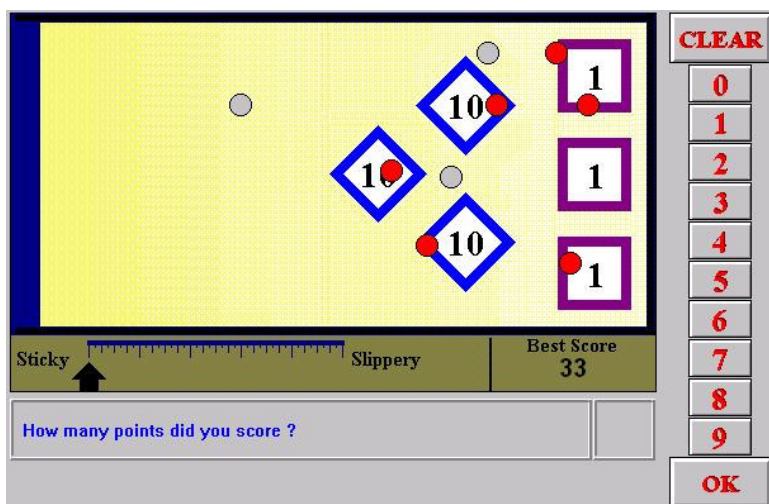


Figura 5.8 – Final de uma partida no Shuffle Board.

Caso o usuário erre na contagem dos pontos, o jogo demonstra como obter a resposta correta através do uso de uma animação que totaliza o valor das placas. O *feedback* fornecido pelo software utiliza mensagens, animações e efeitos sonoros, contribuindo para aumentar o interesse do usuário pelo mesmo.

5.1.4 Secret of the Lost City

O Secret of the Lost City é um jogo que tem como tema uma missão envolvendo três personagens: um casal de astronautas e um pequeno robô flutuante que ao serem atingidos por projéteis de uma nave inimiga, caem num planeta desconhecido. Para consertar sua nave e sair do planeta, o jogador deve resolver problemas aritméticos em 4 fases distintas.

Ao ser iniciado, o jogo solicita o nome do usuário. Após entrar o nome, o aprendiz é levado a uma tela inicial de onde ele pode escolher um entre 5 jogos (Figura 5.9).



Figura 5.9 – Tela inicial do jogo.

Durante cada uma das fases, o usuário tem de completar, selecionar ou montar expressões aritméticas. Os personagens, guiados pelas ações de um usuário, ajudam na seleção de números e equações (Figura 5.10).

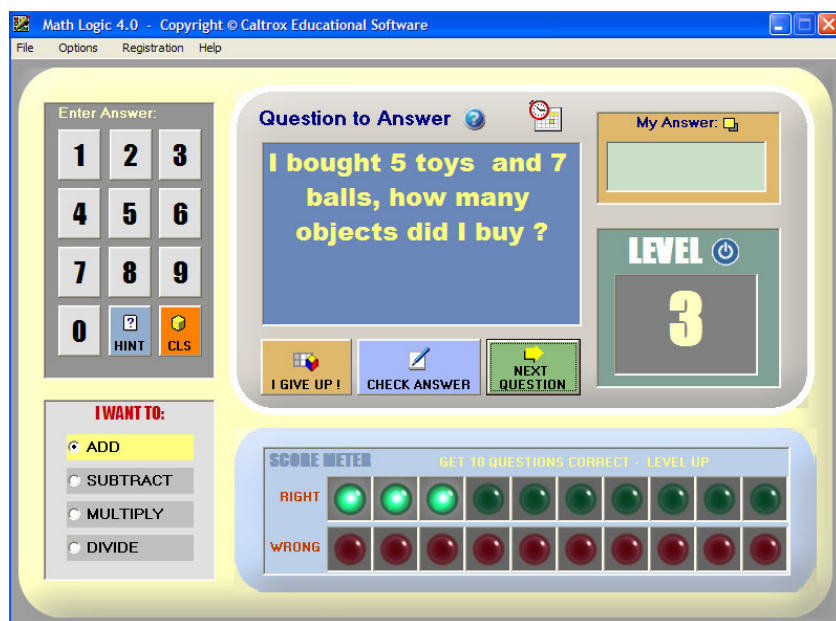


Figura 5.11 – Problema exibido pelo Math Logic.

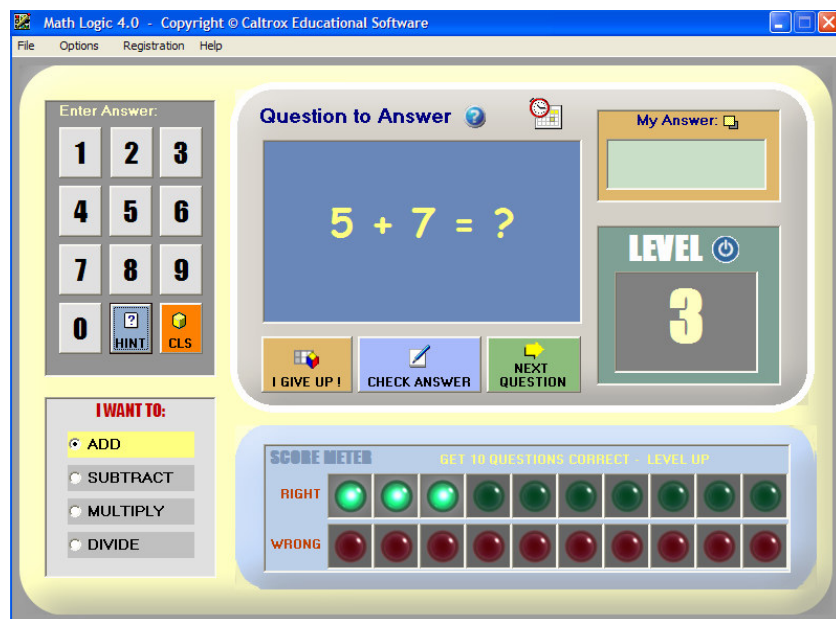


Figura 5.12 – Substituição do enunciado pela dica.

Ao responder uma questão, o programa marca um ponto na linha correspondente a acertos ou erros. Ao acertar dez problemas, o computador informa o percentual de acertos do aluno e aumenta o nível de dificuldade das perguntas. É possível configurar o nível de dificuldade ao iniciar o uso do programa, o que irá alterar a ordem de grandeza dos números apresentados nos problemas.

5.1.6 Comentários sobre os competidores

A análise dos competidores mostrou a existência de produtos que oferecem formas alternativas de apresentar o conteúdo educacional aos usuários. Cada um deles trouxe à tona, seja por méritos ou por deficiências, requisitos importantes para interfaces educativas.

A preocupação em esclarecer o uso da interface bem como fornecer *feedback* ao usuário é visível no Ahsha Math. A presença de um agente que se movimenta na tela e fala, chamando o usuário pelo nome, contribui para facilitar a interação entre o usuário e a aplicação bem como torna o software atrativo para crianças. Outro recurso interessante oferecido por esse software é a possibilidade de configurar os problemas que serão apresentados ao aluno, permitindo ao professor criar atividades de acordo com o conteúdo sendo ministrado em sala de aula.

De acordo com Soloway [SOL94], entre as necessidades básicas dos aprendizes que deve ser contemplada em um software educacional está a motivação. Tornar um software interessante para o usuário facilita obter do mesmo a concentração e dedicação necessárias à aprendizagem do assunto veiculado na interface. A apresentação do conteúdo educacional na forma de jogo é um meio comum de atender a esta necessidade específica dos aprendizes.

Três das interfaces analisadas (Paddle Ball, Shuffle Board e Secret of the Lost City) se apresentam no formato de jogo de forma a atrair a atenção dos usuários. A análise desses programas mostrou a necessidade de uma integração adequada entre o conteúdo educacional e o enredo do jogo. A importância desse requisito apareceu de forma clara no Paddle Ball, devido à separação existente entre esses dois componentes neste software. A interrupção da partida apenas para apresentar um problema para o usuário resolver, sem nenhuma conexão com o desenrolar do jogo, se torna um inconveniente para o aluno, que pode decidir responder qualquer coisa apenas para retornar mais rápido ao jogo.

Software educacional necessita de vários requisitos não funcionais. Clareza na exposição do problema, possibilidade de corrigir erros bem como a presença de instruções sobre a ação a ser realizada em um dado instante são requisitos cuja importância fica clara no momento em que esses recursos estão ausentes. A necessidade de um software educacional permitir a correção de erros acidentais aparece com força no Secret of the Lost City. Neste software, a performance exigida dos usuários para impedir que os bonecos que carregam os valores errados cheguem ao nível inferior torna a ocorrência de erros acidentais um evento comum. A impossibilidade de desfazer uma ação realizada por acidente ou engano pode levar à frustração do usuário com o software, reduzindo o seu interesse no mesmo.

A facilidade de instalar e utilizar um software é um requisito importante para todo aplicativo nos dias atuais. No caso de software educacional ele se torna ainda mais importante, devido a diversidade de computadores e sistemas operacionais que podem ser

encontrados em um laboratório de informática de uma escola. Isto ficou claro durante a realização desta etapa do nosso trabalho, devido à dificuldade em instalar alguns dos aplicativos educacionais utilizados neste trabalho, tendo sido necessária a utilização de uma versão antiga do sistema operacional Windows para a realização dos testes.

Quanto ao conteúdo educacional, percebemos nas interfaces analisadas a presença de poucas situações para o aprendizado das estruturas aditivas. A quantidade de representações disponíveis era também escassa, com os aplicativos se limitando a exigir do usuário a realização de uma operação matemática. O *scaffold* utilizado nos softwares testados, à exceção do Paddle Ball e do Shuffle Board, se limitava a um *feedback* que informava se o valor fornecido pelo usuário para a equação estava certo ou errado. Esses resultados demonstram um distanciamento entre essas interfaces e as pesquisas relacionadas ao aprendizado das estruturas aditivas, que enfatizam a necessidade do uso de diferentes situações e representações para o aprendizado dos conceitos presentes no campo conceitual aditivo.

O quadro 5.1 resume a discussão sobre a aprendizagem do campo conceitual aditivo a partir das interfaces analisadas, apresentando uma comparação entre os produtos vistos sob a ótica da teoria dos campos conceituais.

Quadro 5.1 – Comparação entre os competidores.

Software	Ahsha Math	Paddle Ball	Shuffle Board	Secret of The Lost City	Math Logic
Tipo de Software	Aplicativo	Jogo	Jogo	Jogo	Aplicativo
Uso de situações	Não	Sim	Sim	Não	Sim
Situações utilizadas	----	Composição com total desconhecido; Transformação com total desconhecido.	Composição com total desconhecido.	----	Composição com total desconhecido; Transformação com total desconhecido.
Quantidade de representações	1	2	1	1	2
Tipo de representações Utilizadas	Equação algébrica	Equação algébrica / Material concreto	Uso de placas com dezenas e unidades.	Equação algébrica	Equação algébrica / Enunciado do problema
Associação entre diferentes representações	Não	Não	Não	Não	Não
Uso de <i>scaffolding</i>	Não	Sim (Demonstração)	Sim (Demonstração)	Não	Não

5.1.7 Requisitos levantados a partir da análise de competidores

A partir da observação do comportamento e funcionamento das aplicações escolhidas para a análise dos competidores, foi possível obter requisitos para o nosso software educacional. A seguir apresentamos a lista de requisitos obtidos nesta etapa.

[REQUISITO_01] O software deve fornecer *feedback* constante para o usuário sobre suas ações.

O *feedback* é uma forma eficaz de informar ao usuário o recebimento de um comando bem como torná-lo ciente da avaliação feita pelo software da ação realizada. Quando bem utilizado, o *feedback* pode facilitar a interação do usuário com a interface bem como contribuir para tornar o uso do software mais agradável.

[REQUISITO_02] O conteúdo educacional deve estar integrado a proposta do programa, não sendo oferecido como algo isolado.

A integração entre o conteúdo educacional e a proposta do programa é um requisito importante para a aceitação do software educativo pelo usuário. Isso se torna mais crítico no projeto de jogos educacionais, nos quais a separação do conteúdo educacional do enredo do jogo pode influenciar na motivação do usuário para o aprendizado, como comentado anteriormente em relação ao jogo Paddle Ball.

[REQUISITO_03] O software deve ser compatível com diferentes plataformas computacionais.

A diversidade de plataformas computacionais bem como de sistemas operacionais existente nos laboratórios das instituições de ensino e residência dos usuários exige da aplicação educacional flexibilidade para se adequar aos recursos disponíveis nos ambientes onde a mesma será utilizada.

Uma forma de se obter isso é a utilização, durante o desenvolvimento, de ferramentas que viabilizem a construção de programas multiplataformas, como a linguagem Java.

[REQUISITO_04] O software deve permitir ao professor configurar o mesmo para gerar atividades compatíveis com o programa do curso.

Esse requisito está relacionado à necessidade do software de se adaptar às restrições de uma atividade educacional realizada em um ambiente escolar bem como ao conteúdo programático de um curso de forma a viabilizar o seu uso por parte de instituições de ensino.

[REQUISITO_05] O software deve permitir ao aluno desfazer ações realizadas com o mesmo.

Esse requisito está relacionado à necessidade do aluno de desfazer ações realizadas por engano com o software, bem como fornecer ao mesmo segurança para testar hipóteses e explorar possibilidades durante o aprendizado.

[REQUISITO_06] O software deve permitir a configuração das atividades, visando adaptá-las ao nível do usuário.

Este requisito está relacionado as diferenças em experiência e conhecimento existentes entre os possíveis usuários de um software educativo. Ao permitir a configuração das atividades, o software torna possível o seu uso por usuários iniciantes e experientes.

[REQUISITO_07] O software deve disponibilizar na tela continuamente instruções sobre a ação que o aprendiz deve realizar naquele momento.

Este requisito está relacionado à necessidade do usuário de poder se situar durante a realização de uma atividade com um software educacional. Para isto, a aplicação deve manter acessível ao usuário informações sobre a ação esperada bem como de que forma os componentes da interface podem ser utilizados para realizar esta ação.

[REQUISITO_08] O software deve conter um sistema de ajuda disponível em qualquer momento para esclarecer dúvidas.

Este requisito está relacionado à necessidade do usuário de esclarecer dúvidas sobre o assunto bem como sobre o uso da interface em qualquer momento da utilização do software educacional.

[REQUISITO_09] A ajuda disponibilizada na forma de scaffold pelo software deve preferencialmente fazer uso de recursos multimídia (animações, sons).

Este requisito está relacionado com a necessidade de motivar o usuário para o uso do software educacional. O uso de recursos multimídia, quando bem utilizado, pode tornar o uso do software educacional uma experiência agradável, contribuindo para atrair a atenção do usuário para o conteúdo veiculado na interface.

[REQUISITO_10] O software deve oferecer a possibilidade de registrar as atividades realizadas pelo aluno com o mesmo.

Este requisito está relacionado com a necessidade do professor avaliar o aprendizado do conteúdo educacional por parte dos seus alunos, permitindo ao mesmo

utilizar o *feedback* fornecido pelo sistema sobre o desempenho dos aprendizes na preparação das aulas seguintes. A possibilidade de verificar seu próprio desempenho também deve estar disponível para o aluno, de forma a permitir ao mesmo analisar o seu desempenho na matéria e os pontos a serem melhorados.

[REQUISITO_11] O software deve viabilizar a comunicação entre o professor e os alunos.

Este requisito está relacionado à necessidade do professor se comunicar com os alunos para passar informações importantes sobre a realização de uma atividade com o software. Este deve então permitir que o professor cadastre informações relativas a atividade a ser realizada com a aplicação.

5.2 Prototipação

Como mencionado anteriormente, o protótipo de baixa fidelidade desenvolvido neste trabalho para a realização dos testes com os usuários teve como objetivo auxiliar o aprendizado das estruturas aditivas através da resolução de situações-problemas relacionadas a este campo conceitual. Desta forma, o design do protótipo foi largamente influenciado pelas pesquisas realizadas por Vergnaud e outros pesquisadores sobre este campo conceitual e no uso dos diagramas como forma de levar o aluno a interpretar o problema antes de resolvê-lo.

Uma vez que a finalidade do protótipo é viabilizar a realização dos testes com os usuários antes de construir a versão definitiva do software, apenas as partes do protótipo necessárias à utilização pelos usuários foram criadas nesta etapa. Desta forma, requisitos obtidos na análise dos competidores relacionados às necessidades do professor que utiliza este software em sala de aula com seus alunos não foram prototipados. A razão disso é que esses recursos não são utilizados pelos alunos em sala de aula, e portanto não foram alvo de nossa investigação. Eles estarão presentes, entretanto, na versão final do software.

O protótipo foi criado no computador com o auxílio do software Microsoft Paint, tendo sido impresso e recortado para uso na avaliação.

5.2.1 Descrição da interface do protótipo

A tela principal do protótipo criado pode ser vista na figura a seguir (Figura 5.13):












Figura 5.13 – Tela principal do Gerard.

A interface do protótipo é dividida em três partes principais. A primeira parte (1) contém a seção de comandos, composta pela barra de tarefas e pelos menus. A segunda parte é destinada ao enunciado do problema. Por fim, a terceira parte é destinada a criação e manipulação dos diagramas.

Todas as funções disponibilizadas pelo protótipo poderiam ser acessadas utilizando os menus ou pelos botões da barra de tarefas. As funções disponibilizadas na barra de tarefas podem ser vistas a seguir (quadro 5.2).

Quadro 5.2 – Botões da barra de tarefas.

Botão	Função
 Novo	Permite definir novos problemas, configurar as mensagens e a ajuda com material concreto. Não foi utilizado no teste com o usuário.
 Abrir	Permite carregar novos problemas para serem resolvidos. Não foi utilizado no teste com o usuário.
 Salvar	Grava os modelos criados pelo usuário para cada problema. Não foi utilizado no teste com o usuário.
 Composição	Insere na área de trabalho uma estrutura do tipo composição.
 Comparação	Insere na área de trabalho uma estrutura do tipo comparação.

 Transformação	Insere na área de trabalho uma estrutura do tipo Transformação.
 Selecionar	Seleciona um elemento do diagrama, permitindo alterar o seu valor.
 Refazer	Refaz a última ação realizada.
 Desfazer	Desfaz a última ação realizada.

5.2.2 Descrição do *scaffolding* disponibilizado no protótipo

O projeto do software prevê o uso do programa com ou sem ajuda. No teste foi utilizado o modo com ajuda. A ajuda utilizada em nosso software procurou seguir o modelo proposto por Wood, Bruner e Ross [WOO76] para oferta de *scaffolding*.

No modo com ajuda, apenas os botões da barra de tarefa referentes à escolha da legenda estavam disponíveis para uso por parte dos sujeitos. Essa limitação tinha como objetivo restringir o grau de liberdade do usuário na utilização da interface de forma a viabilizar o controle de suas ações bem como direcioná-lo durante a resolução do problema. Neste modo o sistema monitora as ações do usuário a todo o instante, analisando se as mesmas estão corretas. Caso detecte um erro, o sistema informa ao usuário e fornece uma explicação sobre o erro cometido. A ação considerada errada é rejeitada e o sistema retorna ao estado anterior. Desta forma, apenas ações corretas que aproximem o usuário da solução do problema são aceitas pela aplicação. A rejeição das ações erradas visa impedir o usuário de se desviar do caminho a ser trilhado para obtenção da resposta.

Embora não estivesse disponível no protótipo criado com papel, o sistema foi pensado de forma a aumentar a motivação através do uso de recursos multimídia (animações e sons). Por fim, a demonstração da solução estava disponível na representação com material concreto, caso o usuário não conseguisse resolver o problema utilizando este recurso.

O *scaffolding* disponibilizado no protótipo envolvia:

- Automatização de tarefas;
- Ligação entre as várias representações;
- Uso de representação envolvendo material concreto; e
- *Feedback* para o usuário.

Cada um desses itens será explicado a seguir.

5.2.2.1 Automatização de tarefas

Durante o uso de um software educacional, há interesse em facilitar o aprendizado do usuário iniciante realizando parte do trabalho que deveria ser feito por ele. O usuário é guiado pelo software durante a realização da atividade, e realiza apenas as partes da tarefa que está apto a fazer. Este auxílio é disponibilizado para o usuário no início da utilização do software e deve desaparecer gradualmente à medida que o conhecimento do indivíduo sobre o assunto tratado na interface evolui. Com o desaparecimento desta ajuda, o usuário passa a realizar todas as etapas da tarefa tendo liberdade para escolher o caminho para a resolução do problema, aumentando com isso o grau de liberdade do sujeito na utilização do programa.

Durante o projeto do protótipo procuramos analisar que partes da tarefa do usuário iniciante poderiam ser automatizadas. Consideramos que, para o usuário inexperiente no uso dos diagramas, a montagem e compreensão do mesmo seria um entrave para o uso inicial do programa. Esta dificuldade poderia advir da inexperiência com o uso do software para construção do diagrama ou do desconhecimento do significado dos símbolos utilizados na legenda.

Em nosso protótipo, quando o usuário seleciona uma estrutura (composição, comparação ou transformação) para o problema a ser resolvido, o diagrama com todos os componentes já aparece montado de acordo com as necessidades do problema, cabendo ao usuário iniciante apenas preencher os valores. Por exemplo, no caso do problema 1, ao escolher composição, o usuário já receberia o diagrama com os dois quadrados que representam as partes devidamente posicionados no diagrama (Figura 5.14).

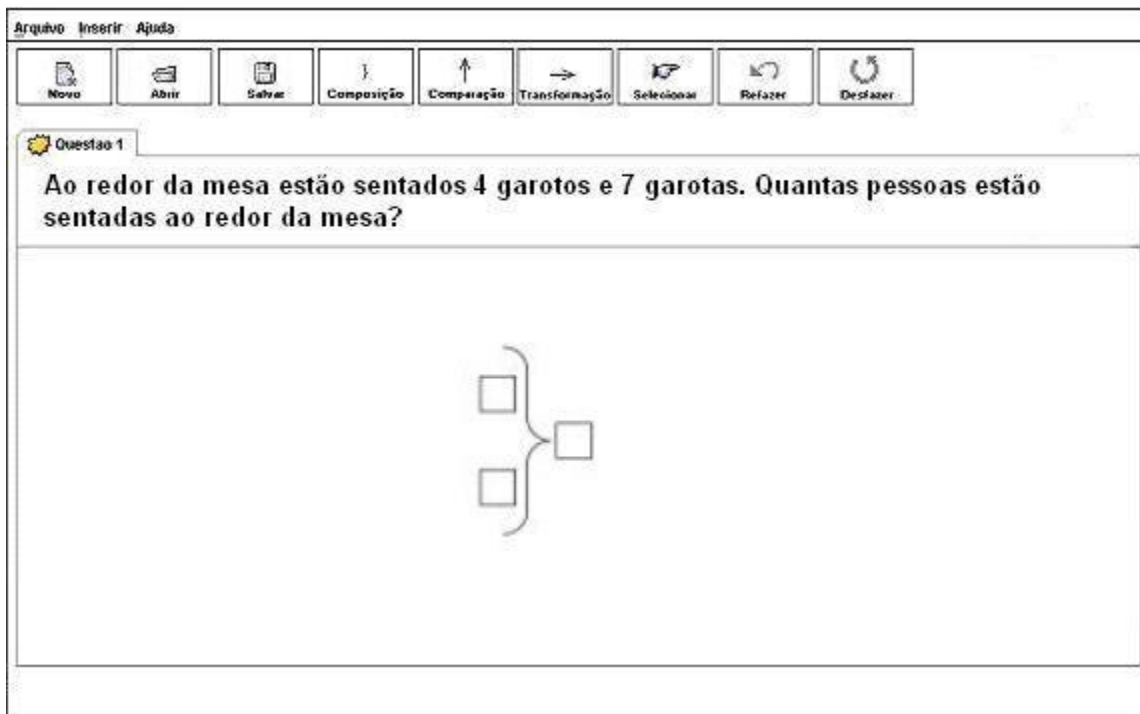


Figura 5.14 – Diagrama de composição para o problema 1.

No modo de utilização no qual esta ajuda não está disponível, a seleção da composição por parte do usuário significará apenas o aparecimento da chave com o quadrado que representa o total de objetos na tela, cabendo ao usuário identificar quantas partes estão presentes no problema e colocar os quadrados manualmente no diagrama.

Outra ajuda disponibilizada foi o uso de texto explicando o significado de cada elemento dentro do contexto do problema. Essa ajuda só era oferecida caso o usuário cometesse erros na montagem do diagrama. Um exemplo dessa ajuda pode ser visto a seguir (Figura 5.15):

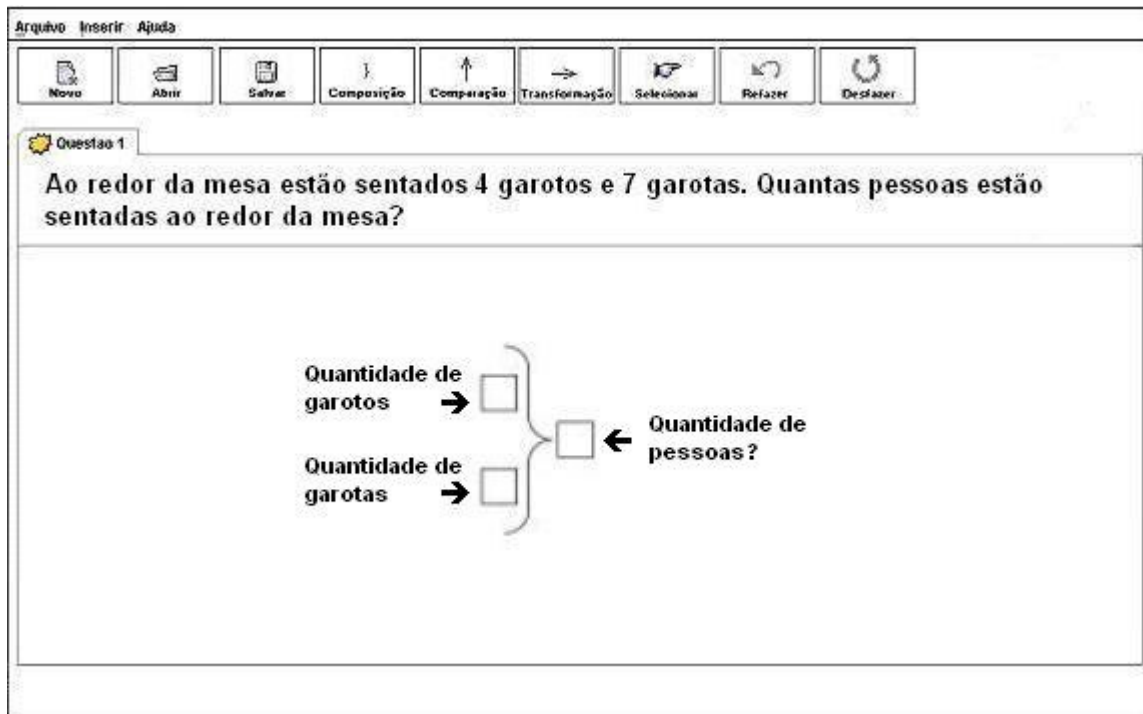


Figura 5.15 - Diagrama de composição com texto explicativo.

5.2.2.2 Ligação entre as várias representações

Como foi visto, a teoria dos campos conceituais enfatiza que um conceito é formado por um conjunto de situações, invariantes e representações. Uma mesma propriedade de um conceito pode ser representada de mais de uma maneira, e é importante que o aprendiz consiga passar de uma representação para outra durante a resolução do problema.

De acordo com Kaput e Kozma, citado por Rick [RIC01], a utilização de múltiplas representações para os conceitos veiculados em um software educacional favorece o aprendizado desses conceitos. Rick [*Ibid*] destaca a importância de conectar as diferentes representações como forma de permitir ao aprendiz explorar diferentes aspectos do domínio trabalhado pelo software.

O protótipo foi projetado de forma a permitir a troca de informações entre uma representação para outra. Números podem ser arrastados do enunciado do problema direto para o diagrama, efetivamente conectando as duas representações (Figura 5.16).

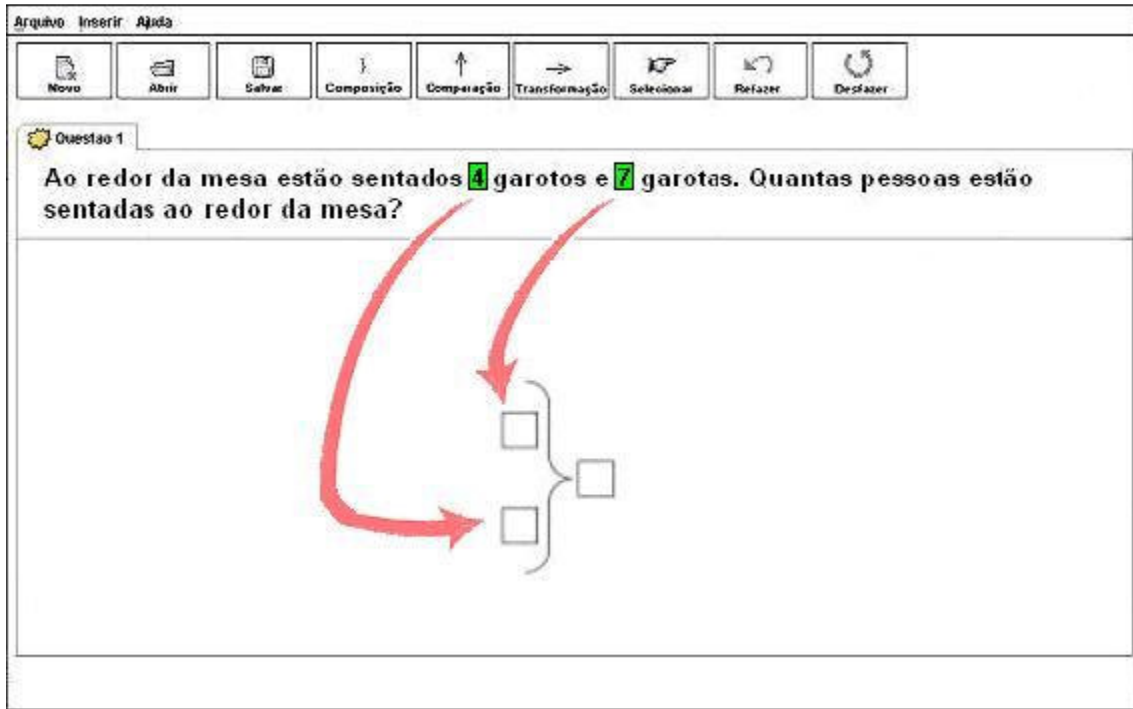


Figura 5.16 – Funcionamento da conexão entre duas representações.

5.2.2.3 Uso de representação envolvendo material concreto

O material concreto é um recurso utilizado largamente como apoio ao ensino da matemática nas séries iniciais [SPI04, p. 7]. O uso de uma metáfora deste recurso em nosso protótipo foi pensado como uma nova representação a qual o usuário poderia recorrer caso tivesse dificuldades na resolução do problema usando o diagrama proposto por Vergnaud. Em nosso protótipo, o usuário manipula objetos que podem ter natureza diferente dos elementos tratados no problema, cabendo ao mesmo relacioná-los dentro do contexto da questão sendo resolvida.

Em nosso protótipo, três tipos de ajuda estavam disponíveis para o usuário. A primeira envolvia a manipulação de conjuntos de objetos, e podia ser utilizada tanto em problemas de composição como transformação. O funcionamento dessa ajuda para o problema 1 (Quadro 5.3) pode ser visto a seguir.

Quadro 5.3 – Problema 1.

Ao redor da mesa estão sentados 4 garotos e 7 garotas. Quantas pessoas estão sentadas ao redor da mesa?

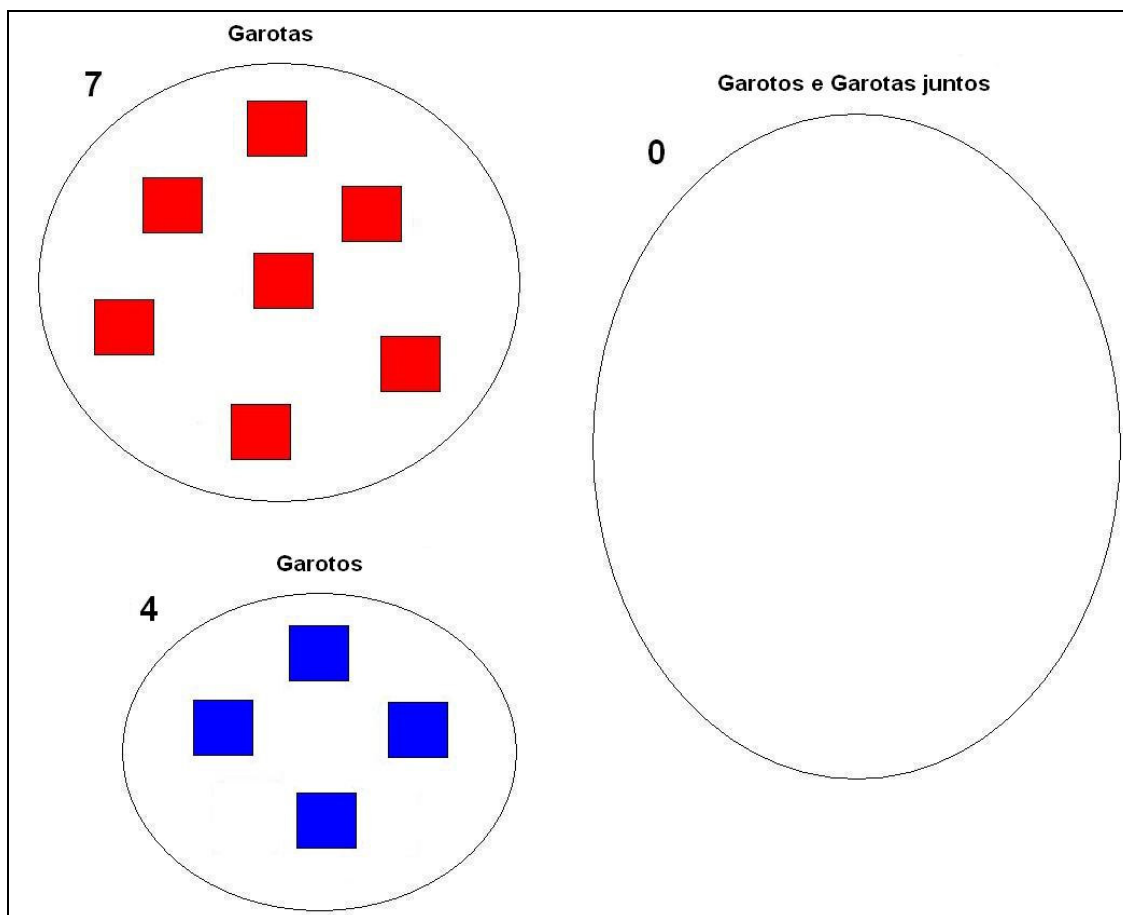


Figura 5.17 – Configuração inicial da ajuda para problemas de composição.

No exemplo mostrado na Figura 5.17, era solicitado que o usuário arrastasse os objetos dos dois primeiros conjuntos da esquerda (garotos e garotas) até o conjunto dos garotos e garotas juntos. À medida que os objetos eram arrastados, o número associado ao cardinal da nova coleção era atualizado. Os objetos provenientes de uma mesma coleção possuem uma cor específica (neste exemplo, vermelho para as garotas e azul para os garotos), de forma a facilitar a identificação da coleção de origem do objeto. Ao final deste exercício, o usuário teria formado um novo conjunto a partir dos objetos das duas coleções iniciais (Figura 5.18). Ao arrastar o último elemento, o usuário era solicitado a responder quantos elementos existem agora no conjunto da direita.

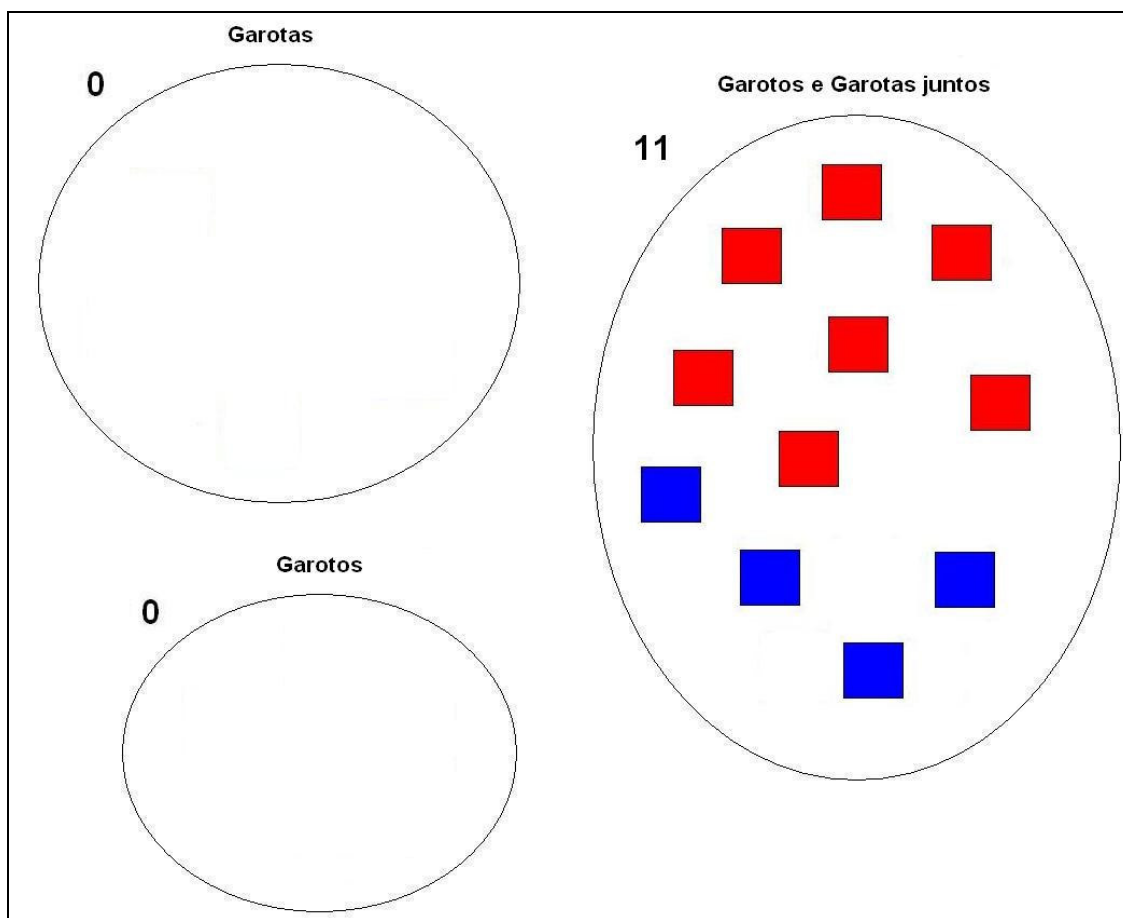


Figura 5.18 - Ajuda para problemas de composição após arrastar os quadrados.

Esta interação tinha como objetivo facilitar a visualização da relação entre o número que representava o cardinal da coleção e a quantidade de objetos disponíveis no conjunto. Ao “construir” a resposta do problema arrastando os quadradinhos, nosso objetivo era criar condições para o aluno perceber o invariante “as partes formam o todo”, que é necessário para a resolução deste problema.

A segunda ajuda estava relacionada exclusivamente a problemas de transformação com estado inicial ou final desconhecido, como no exemplo a seguir (Quadro 5.4).

Quadro 5.4 – Problema 3.

Maria comprou uma caixa de bombons por 4 reais e ainda ficou com 4 reais. Quanto ela possuía antes de fazer a compra?

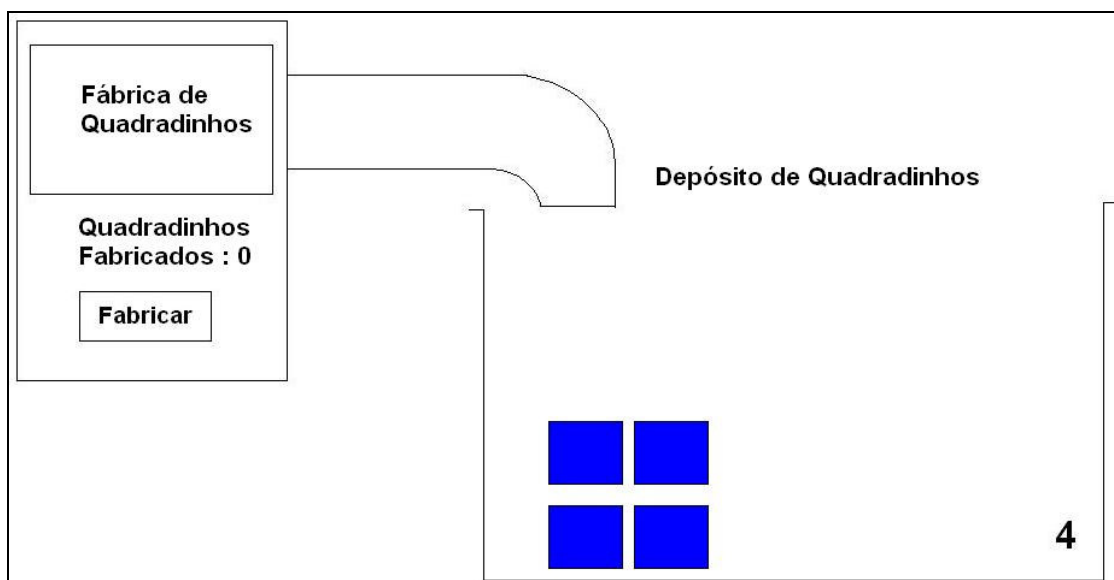


Figura 5.19 – Configuração inicial da ajuda para problemas de transformação.

Para este problema, era utilizada uma ajuda que envolvia uma fábrica de quadrados, sendo solicitado que o usuário fabricasse a quantidade de elementos equivalente à quantidade de reais gasto por Maria (Figura 5.19). A cada clique no botão “Fabricar”, um quadrado de cor vermelha era adicionado ao conjunto de quadrados pré-existente (todos azuis), sendo o valor do número de elementos fabricados atualizado. Após o usuário fabricar a quantidade de quadrados correta (Figura 5.20), era solicitado que o mesmo respondesse quantos elementos havia no depósito.

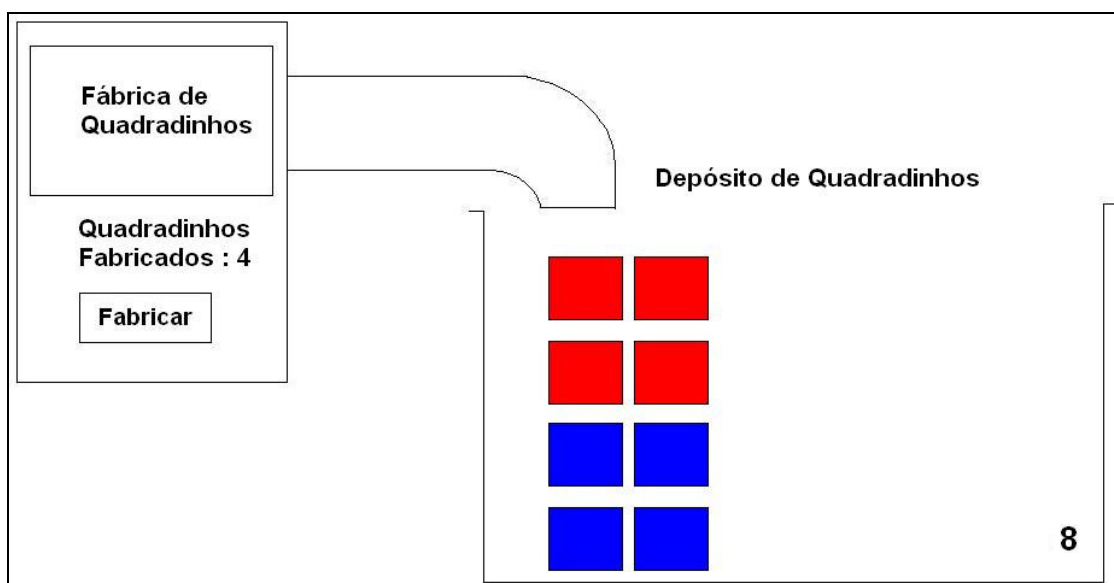


Figura 5.20 - Ajuda para problemas de transformação após fabricar os quadrados.

O objetivo desta ajuda era permitir ao usuário chegar à resposta utilizando um estilo de interação diferente da primeira ajuda, na qual os elementos eram manipulados fisicamente (arrastados). Nesta ajuda, a interação do aluno se limitava a pressionar o

botão fabricar e ver a quantidade de elementos no depósito ser incrementada. À medida que os quadradinhos eram colocados no depósito, o número associado ao cardinal da coleção era atualizado. Ao “construir” a resposta do problema fabricando os quadradinhos, nosso objetivo era criar condições para o aluno perceber o invariante que permite a obtenção do estado inicial de uma quantidade partindo-se do estado final da mesma e adicionando a transformação inversa que produziu esse estado final, invariante este necessário para a resolução deste problema.

A terceira ajuda era destinada aos problemas em que havia uma comparação entre duas quantidades, como no exemplo a seguir (Quadro 5.5).

Quadro 5.5 – Problema 5.

Carlos tem 7 reais e Luiz tem 6 reais a menos do que ele. Quantos reais tem Luiz?



Figura 5.21 - Configuração inicial da ajuda para problemas de comparação.

Na ajuda projetada para este problema (Figura 5.21), era solicitado que o usuário retirasse de Luiz uma quantidade de quadradinhos equivalente à diferença de reais entre ele e Carlos, arrastando esses quadradinhos para o círculo. A cada quadradinho arrastado, o valor no círculo era incrementado e o valor junto à barra diminuído. Após o usuário

arrastar a quantidade de quadradinhos correta, era perguntado quantos quadradinhos restaram na barra de Luiz (Figura 5.22).

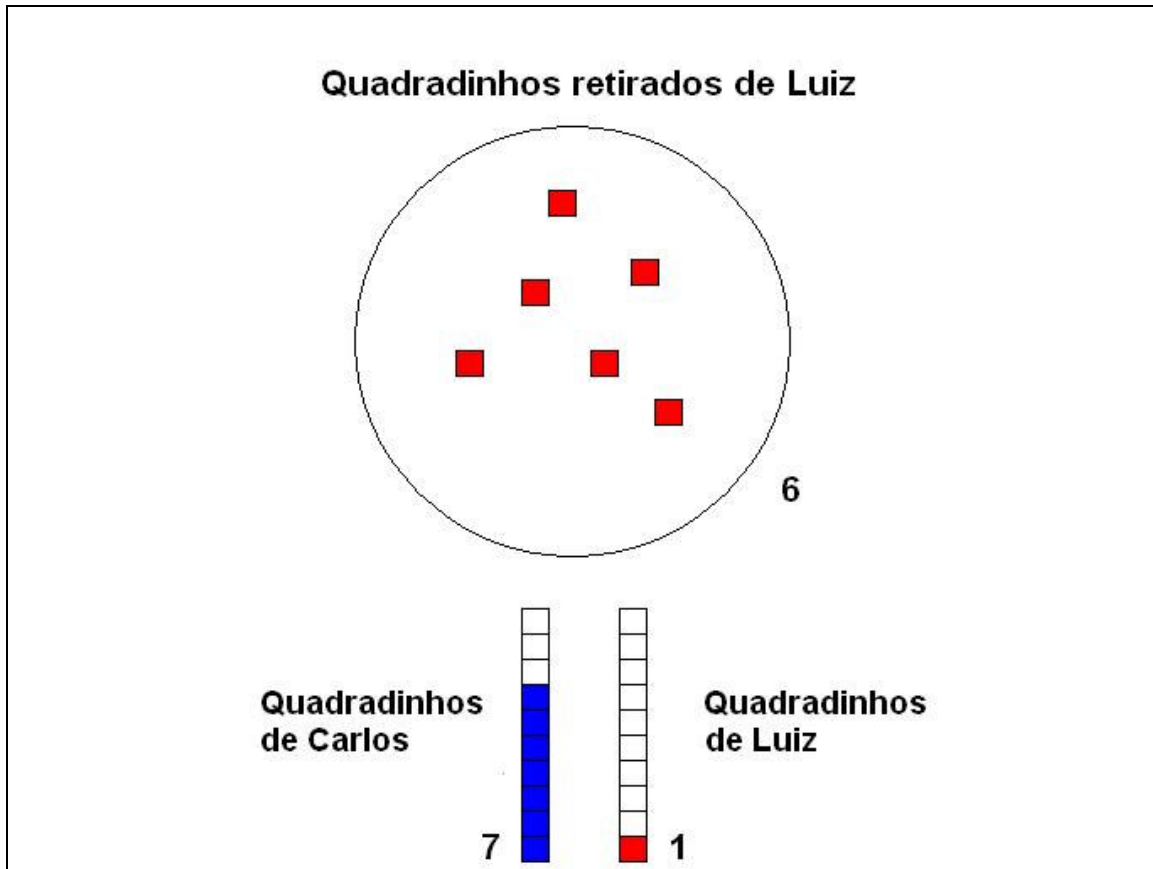


Figura 5.22 - Ajuda para problemas de transformação após a retirada dos quadrados.

Esta interação tinha como objetivo facilitar a obtenção da resposta da questão bem como a percepção por parte do usuário do invariante que permite a obtenção de uma medida especificada em relação a uma outra a partir do valor desta outra medida e da relação existente entre as duas.

5.2.2.4 *Feedback* para o usuário

O *feedback* sobre as ações realizadas é muito importante em um software educacional. Além de orientar o usuário sobre o seu progresso na resolução da tarefa, o *feedback* serve para guiar o usuário na resolução do problema, podendo também aumentar a motivação do usuário em utilizar a interface.

Em nosso protótipo, o *feedback* era frequentemente disponibilizado em forma de mensagens que tinham como objetivo alertar o usuário sobre erros de modelagem, fornecer esclarecimentos sobre o uso do protótipo e do diagrama proposto por Vergnaud bem como disponibilizar a ajuda com o material concreto. O uso das mensagens serviu também para manter o usuário na direção correta para a resolução do problema através da

redução do grau de liberdade (apenas as ações corretas eram aceitas pelo sistema). Um exemplo de uma mensagem pode ser visto a seguir (Figura 5.23).

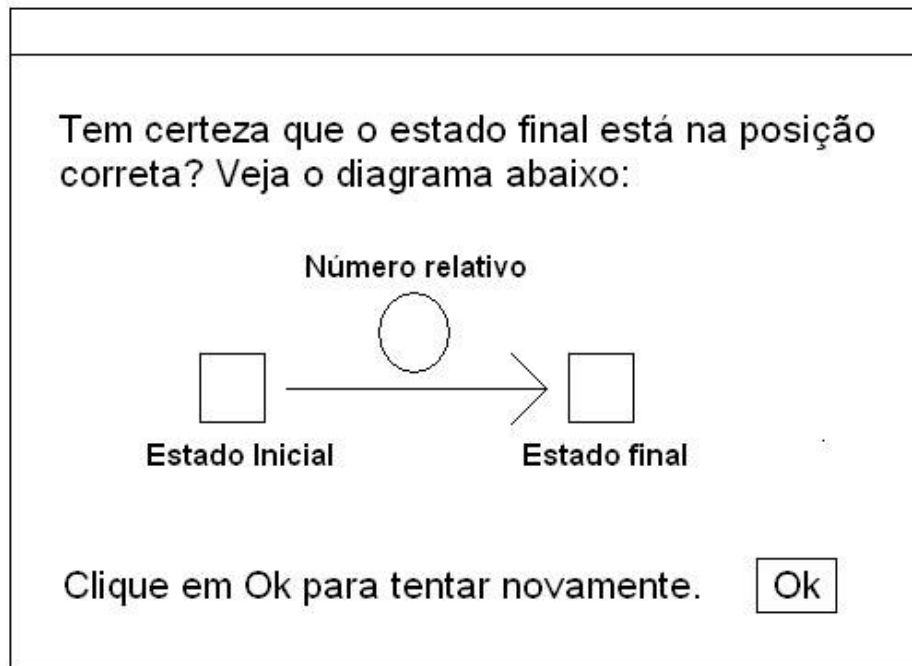


Figura 5.23 – Mensagem utilizada no protótipo.

5.3 Testes de usabilidade

Como foi visto no capítulo anterior, a análise dos dados dos testes teve como objetivo identificar:

- Os invariantes mobilizados pelos usuários no uso do protótipo; e
- A contribuição da ajuda disponibilizada na forma de *scaffolding* ao processo de resolução dos problemas com a interface educacional.

As transcrições dos testes realizados junto aos usuários foram classificadas em categorias de forma a facilitar a visualização dos resultados. As categorias definidas para a análise dos resultados dos testes realizados junto aos usuários podem ser vistas a seguir (Figura 5.24).

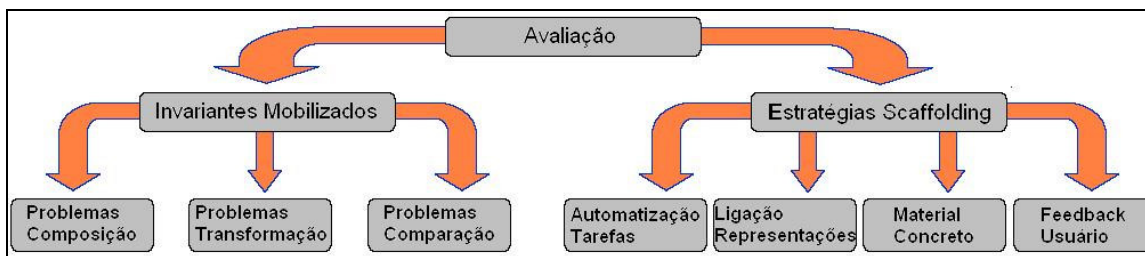


Figura 5.24 - Categorias de análise.

Alguns fragmentos das transcrições serão mostrados para exemplificar os resultados obtidos em cada categoria. Para facilitar o entendimento, as transcrições utilizam alguns sinais, listados a seguir (Quadro 5.6).

Quadro 5.6 – Sinais utilizados nas transcrições.

Sinal	Descrição
S	Diálogo do sujeito.
P	Diálogo do pesquisador.
C	Diálogo do computador.
()	Comentário do analista.
[]	Descrição do problema sendo resolvido pelo usuário.

5.3.1 Análise dos invariantes mobilizados pelos sujeitos

Nesta categoria separamos os invariantes observados durante os testes de acordo com o tipo de problema: composição, comparação e transformação. Ao todo, um total de 13 invariantes corretos foram observados durante a realização dos testes. Os invariantes incorretos não foram objeto de análise, uma vez que investigar as razões que levam o usuário a elaborar uma estratégia incorreta para a resolução de um problema foge ao escopo desse trabalho.

É importante ressaltar que o fato do usuário não ter mencionado o invariante na explicação de uma ação não quer dizer necessariamente que ele não fez uso dele. Um indivíduo pode fazer uso de um raciocínio sem estar consciente disso [MOR02]. Apesar disso acreditamos que, para que a aprendizagem possa ocorrer de forma consistente, o usuário deve estar sempre ciente daquilo que ele realizou ao utilizar um programa educacional. É nosso interesse então que o usuário perceba que raciocínio o mesmo lançou mão ao utilizar o software. Por esta razão, em nossa pesquisa, quando o usuário não deixou claro porque realizou uma determinada ação consideramos o pior caso, ou seja, aqueles usuários que não mencionaram de forma clara um dado invariante não fizeram uso dele.

5.3.1.1 Invariantes mobilizados na resolução de problemas de composição

Como visto no capítulo anterior, problemas de composição são aqueles em que são apresentadas situações que envolvem parte – todo, isto é, juntar uma parte com outra parte para obter o todo, ou subtrair uma parte do todo para obter a outra parte [MAG02].

A análise da transcrição dos dados do experimento com o uso do protótipo permitiu a identificação de 3 invariantes mobilizados pelo sujeito durante a resolução dos problemas de composição. Cada um desses invariantes é analisado a seguir.

Invariante 1: As partes formam o todo

Este invariante apareceu no discurso do usuário no momento em que o mesmo identificava o problema como uma composição de duas quantidades. A percepção da situação apresentada na questão como uma composição se traduziu no reconhecimento por parte do sujeito de que existem duas coleções de objetos (as partes) cujo total pode ser obtido a partir da soma do cardinal de cada coleção separadamente.

Este raciocínio foi detectado nos discursos de S1, S3 e S4. Um exemplo da presença desse invariante pode ser vista na explicação de S1 para a escolha da composição para o problema 1 (Quadro 5.7).

Quadro 5.7 – Fragmento 1 do teste.

FRAGMENTO 1 : PROBLEMA 1 : SUJEITO 1
[Ao redor da mesa estão sentados 4 garotos e 7 garotas. Quantas pessoas estão sentadas ao redor da mesa?]
C: Escolha no menu a legenda correspondente à operação.
S: Composição (clicou no botão composição).
P: Porque você escolheu composição para esse problema?
S: Na verdade, porque o problema indica que há pessoas de sexo masculino e feminino, e pergunta quantas pessoas existem. Nesse sentido tem que somar.

Nem sempre o usuário utiliza esse invariante corretamente. Ao se deparar com um problema de transformação, S3 identificou a questão erroneamente como uma composição. Ao ser questionado sobre a sua escolha, o sujeito forneceu a seguinte explicação (Quadro 5.8):

Quadro 5.8 – Fragmento 2 do teste.

FRAGMENTO 2 : PROBLEMA 3 : SUJEITO 3
[Maria comprou uma caixa de bombons por 4 reais e ainda ficou com 4 reais. Quanto ela possuía antes de fazer a compra?] P: Porque você escolheu composição para esse problema? S: Bom, na transformação, da mesma forma que na composição, você também está operando com duas quantidades. Se eu tenho (um exemplo fornecido pelo usuário) 4 bolas e ganhei 3 bolas, também se trata de uma composição. Então os problemas de composição e de transformação são muito parecidos.

S3 voltou a cometer o mesmo erro posteriormente, identificando como uma composição um problema de comparação (problema 4). Questionado sobre a sua escolha, o sujeito forneceu a seguinte explicação (Quadro 5.9).

Quadro 5.9 – Fragmento 3 do teste.

FRAGMENTO 3 : PROBLEMA 4 : SUJEITO 3
[Ricardo tem 6 anos, Carlos tem 4 anos a mais do que ele. Quantos anos tem Carlos?] P: Porque você escolheu composição para esse problema? S: Porque no problema eu estou juntando a idade de Ricardo com os 4 anos que Carlos tem a mais. Então seria uma junção de duas quantidades.

Este evento mostrou a necessidade de fornecer, dentro do software, uma ajuda mais detalhada relacionada a este tipo de erro, no qual o usuário tem dificuldades em identificar a estrutura trabalhada no problema. Veremos essa questão em detalhes na seção 5.4.

Invariante 2: Um número pode representar o cardinal de um conjunto

Este invariante está relacionado ao sentido atribuído ao número por parte do sujeito. O indivíduo reconhece no número uma representação conveniente para o cardinal de uma coleção de objetos, percebendo a relação existente entre o número no enunciado do problema e o total de objetos em uma coleção.

Esse invariante foi detectado no discurso de todos os participantes durante a resolução do problema envolvendo uma composição. Um exemplo pode ser visto no discurso de S2 como visto a seguir (Quadro 5.10).

Quadro 5.10 – Fragmento 4 do teste.

FRAGMENTO 4 : PROBLEMA 1 : SUJEITO 2
[Ao redor da mesa estão sentados 4 garotos e 7 garotas. Quantas pessoas estão sentadas ao redor da mesa?]
P: Você arrastou o número 4 para este quadrado (uma parte). Porque você fez isso?
S: Eu coloquei aqui porque nos quadrados deste lado (lado esquerdo da composição) deveria ficar a quantidade de garotos, que é 4, e a quantidade de garotas, que é 7.
P: Você arrastou o número 7 para este outro quadrado (a outra parte). Porque você fez isso?
S: É a mesma coisa. O 7 é a quantidade de garotas e deve ficar deste lado (lado esquerdo da composição).

Invariante 3: A soma das quantidades gera um valor que corresponde à cardinalidade do todo

Este invariante está relacionado com a percepção por parte do sujeito de que o cardinal da união de dois conjuntos pode ser obtido pela soma do cardinal de cada coleção individualmente. A percepção deste invariante é fundamental para a resolução do problema, e por esta razão, é muito importante que o software leve o usuário a mobilizar este conhecimento.

Esse raciocínio foi observado no discurso de S1, S3 e S4. Um exemplo pode ser visto no discurso de S3 como visto a seguir (Quadro 5.11).

Quadro 5.11 – Fragmento 5 do teste.

FRAGMENTO 5 : PROBLEMA 1 : SUJEITO 3
[Ao redor da mesa estão sentados 4 garotos e 7 garotas. Quantas pessoas estão sentadas ao redor da mesa?]
P: Você indicou o valor 11 para este quadrado (o todo). Como você explicaria isso?
S: Você tem dois conjuntos, e o problema requer que você indique o valor do conjunto formado pela união dos dois conjuntos. Adicionando o número de elementos de cada um desses conjuntos, o resultado então seria 11.

Como foi dito anteriormente, acreditamos que é preciso que o usuário torne claro o que o levou a realizar determinada ação. Uma ação correta sem a devida justificativa não pode ser considerada como resultado da utilização correta de um invariante, sob pena de incorrer em um erro. O problema 1 também foi resolvido por S2. Quando questionado do porque da realização da última ação, o sujeito não conseguiu fornecer uma explicação que permitisse observar o raciocínio utilizado (Quadro 5.12):

Quadro 5.12 – Fragmento 6 do teste.

FRAGMENTO 6 : PROBLEMA 1 : SUJEITO 2
[Ao redor da mesa estão sentados 4 garotos e 7 garotas. Quantas pessoas estão sentadas ao redor da mesa?]
P: Você indicou o valor 11 para este quadrado (o todo). Como você explicaria isso?
S: Porque eu somei $4 + 7$.

Ao mencionar apenas o cálculo matemático realizado, o sujeito impede que seja conhecido que invariante o mesmo utilizou na resolução do problema. O uso consciente dos invariantes é um objetivo que pretendemos alcançar na construção do software. Por essa razão consideramos que o invariante 3 não foi mobilizado por S2 na resolução do problema 1.

5.3.1.2 Invariantes mobilizados na resolução de problemas de transformação

Os problemas de transformação envolvem situações nas quais a idéia de tempo está sempre presente. Neste tipo de problema, uma medida sofre uma transformação através de perda ou ganho, dando como resultado uma nova medida.

Invariante 4: Uma medida inicial se transforma, dando origem a uma nova medida.

Este invariante apareceu no discurso do usuário no momento em que o mesmo identificava o problema como uma situação envolvendo a transformação de uma medida. A percepção da situação apresentada na questão como uma transformação exigiu do sujeito o reconhecimento de que existe uma medida que possui dois valores distintos, cada um correspondendo a um instante de tempo diferente. Estes dois valores estão relacionados pela transformação de perda ou ganho que foi aplicada à medida entre esses dois instantes de tempo.

Este raciocínio foi detectado nos discursos de S1, S2 e S4. Um exemplo da presença desse invariante pode ser vista na explicação de S4 para a escolha da transformação para o problema 2 (Quadro 5.13).

Quadro 5.13 – Fragmento 7 do teste.

FRAGMENTO 7 : PROBLEMA 2 : SUJEITO 4
[Ricardo saiu de casa para jogar bola de gude. Ao sair de casa ele possuía 6 bolas. Após o jogo ele tinha duas bolas. O que aconteceu no jogo?]
P: Porque você escolheu transformação para esse problema?
S: Porque o problema informa a quantidade de bolas que Ricardo possuía antes e após o jogo, e pergunta o que ocorreu no jogo. Então temos uma transformação da quantidade inicial de bolas que ele tinha na quantidade final.

Invariante 5: Um número pode representar o estado inicial da medida.

Este invariante está relacionado ao sentido atribuído ao número por parte do sujeito. Em problemas de transformação, o indivíduo reconheceu no número uma representação conveniente para uma medida em seu estado inicial, antes de sofrer a modificação que irá levá-la ao estado final.

Este raciocínio foi detectado nos discursos de S1, S3 e S4 ao justificar a indicação de um número do enunciado para o quadrado que representa o estado inicial ao resolver um problema de transformação. Um exemplo pode ser visto no discurso de S1 como visto a seguir (Quadro 5.14).

Quadro 5.14 – Fragmento 8 do teste.

FRAGMENTO 8 : PROBLEMA 2 : SUJEITO 1
[Ricardo saiu de casa para jogar bola de gude. Ao sair de casa ele possuía 6 bolas. Após o jogo ele tinha duas bolas. O que aconteceu no jogo?]
P: Você arrastou o número 6 para este quadrado (o estado inicial). Porque você fez isso?
S: Porque o ponto de partida do problema é o número 6, na verdade no início Ricardo tinha 6 bolas.

Invariante 6: Um número pode representar o estado final da medida.

Este invariante é semelhante ao anterior, estando relacionado ao sentido atribuído ao número por parte do sujeito. Neste caso, o número representa a medida após sofrer a transformação.

Este raciocínio também foi detectado nos discursos de S1, S2, S3 e S4 ao justificar a indicação de um número do enunciado para o quadrado que representa o estado final em problemas de transformação. Um exemplo pode ser visto no discurso de S4 como visto a seguir (Quadro 5.15).

Quadro 5.15 – Fragmento 9 do teste.

FRAGMENTO 9 : PROBLEMA 2 : SUJEITO 4
[Ricardo saiu de casa para jogar bola de gude. Ao sair de casa ele possuía 6 bolas. Após o jogo ele tinha duas bolas. O que aconteceu no jogo?]
P: Você arrastou o número 2 para este quadrado (o estado final). Porque você levou o número dois para esse quadrado?
S: Porque o dois é a quantidade de bolas que Ricardo possui após terminar o jogo.

Invariante 7: A transformação aplicada a uma medida pode ser expressa por um número.

Este raciocínio foi detectado nos discursos de S1, S2, S3 e S4 ao justificar a indicação de um número como a transformação que leva o estado inicial ao estado final. Um exemplo pode ser visto no discurso de S3 como visto a seguir (Quadro 5.16).

Quadro 5.16 – Fragmento 10 do teste.

FRAGMENTO 10 : PROBLEMA 3 : SUJEITO 3
[Maria comprou uma caixa de bombons por 4 reais e ainda ficou com 4 reais. Quanto ela possuía antes de fazer a compra?] P: Você levou o número 4 (dinheiro que Maria gastou) para o círculo (número relativo). Porque você fez isso? S: Porque esse valor representa uma subtração que devo fazer da quantia inicial que ela tinha.

Invariante 8: A transformação aplicada a uma medida é obtida pela diferença entre o estado final e o estado inicial da mesma.

Este raciocínio foi detectado nos discursos de S1, S3 e S4 ao justificar a indicação de um número como a transformação que leva o estado inicial ao estado final. Um exemplo pode ser visto no discurso de S1 como visto a seguir (Quadro 5.17).

Quadro 5.17 – Fragmento 11 do teste.

FRAGMENTO 11 : PROBLEMA 3 : SUJEITO 1
[Maria comprou uma caixa de bombons por 4 reais e ainda ficou com 4 reais. Quanto ela possuía antes de fazer a compra?] P: Você levou o valor -4 no círculo (número relativo). Porque você fez isso? S: Bom, porque seria o número que separava os dois valores, no caso o que Maria possuía e o que Maria tem agora. Então esse valor teria que aparecer no círculo.

Invariante 9: O estado inicial em um problema de transformação pode ser obtido a partir da aplicação da transformação inversa ao estado final da medida.

Este raciocínio foi observado nos discursos de S1, S2, S3 e S4 ao explicar como foi obtido o estado inicial no problema 3. O estado inicial de uma medida pode ser obtido a partir do estado final, desfazendo o efeito produzido pela transformação sobre o estado inicial. Um exemplo desse invariante pode ser visto no discurso de S1 como visto a seguir (Quadro 5.18).

Quadro 5.18 – Fragmento 12 do teste.

FRAGMENTO 12 : PROBLEMA 3 : SUJEITO 1
[Maria comprou uma caixa de bombons por 4 reais e ainda ficou com 4 reais. Quanto ela possuía antes de fazer a compra?]
P: Você colocou o número 8 neste quadrado (estado inicial). Como você explicaria essa ação?
S: Na verdade eu fiz a operação para trás, partindo do dinheiro que Maria tem agora e somei o valor que ela gastou. Assim cheguei ao valor que ela possuía antes, no caso 8, e então coloquei esse valor como resultado.

5.3.1.3 Invariantes mobilizados na resolução de problemas de comparação

A classe de problemas de comparação se refere a problemas que comparam duas quantidades. Nestes problemas temos sempre uma quantidade cujo valor é especificado em relação ao valor de outra quantidade. A primeira quantidade recebe o nome de referido e a segunda recebe o nome de referente.

A análise da transcrição dos dados do experimento onde os usuários resolvem o problema 4 com o uso do protótipo permitiu a identificação de 4 invariantes mobilizados pelo sujeito durante a resolução de problemas de comparação. Cada um desses invariantes é visto a seguir.

Invariante 10: O valor de uma medida é especificado em relação ao valor de uma outra medida, em termos de acréscimos ou decréscimos.

Este invariante apareceu no momento da identificação do problema como uma comparação entre duas quantidades. A percepção da situação apresentada na questão como uma comparação exigiu do sujeito o reconhecimento de que existe uma quantidade cujo valor é especificado em função do valor de outra quantidade e da relação existente entre as duas quantidades.

Este raciocínio foi detectado nos discursos de S1, S2 e S4 quando da escolha da estrutura que melhor modelava um problema de comparação. Um exemplo da presença desse invariante pode ser vista na explicação de S4 para a escolha da composição para o problema 4 (Quadro 5.19).

Quadro 5.19 – Fragmento 13 do teste.

FRAGMENTO 13 : PROBLEMA 4 : SUJEITO 4
[Ricardo tem 6 anos, Carlos tem 4 anos a mais do que ele. Quantos anos tem Carlos?]
P: Porque você escolheu comparação para esse problema?
S: Porque o problema pedia a idade de um garoto, o Carlos, utilizando outro garoto como referência.

Invariante 11: Um número pode representar o referente de uma medida, servindo de base para a obtenção da outra medida.

Este invariante está relacionado ao sentido atribuído ao número por parte do sujeito no contexto do problema. O indivíduo reconhece no número uma representação conveniente para a medida usada como referencial para obter uma outra medida com a qual a mesma está sendo comparada.

Esse invariante foi detectado no discurso de S1, S2, S3 e S4 quando da resolução de problemas de comparação. Um exemplo pode ser visto no discurso de S1 como visto a seguir (Quadro 5.20).

Quadro 5.20 – Fragmento 14 do teste.

FRAGMENTO 14 : PROBLEMA 4 : SUJEITO 1
[Ricardo tem 6 anos, Carlos tem 4 anos a mais do que ele. Quantos anos tem Carlos?]
P: Você arrastou o número 6 para este quadrado (o referente). Como você explicaria isso?
S: O 6 representava a idade a qual deveria ser adicionada uma quantidade de anos para obter a idade do outro sujeito. O 6 então seria o ponto de partida para a solução do problema, seria o referencial.

Invariante 12: A relação existente entre as duas medidas pode ser expressa por um número.

De forma similar ao caso anterior, este invariante está relacionado ao sentido atribuído ao número por parte do sujeito no contexto do problema. O indivíduo reconhece no número uma representação conveniente para a relação existente entre as duas medidas sendo comparadas.

Esse invariante foi detectado no discurso de S1, S2, S3 e S4 quando da resolução de problemas de comparação. Um exemplo pode ser visto no discurso de S2 como visto a seguir (Quadro 5.21).

Quadro 5.21 – Fragmento 15 do teste.

FRAGMENTO 15 : PROBLEMA 4 : SUJEITO 2
[Ricardo tem 6 anos, Carlos tem 4 anos a mais do que ele. Quantos anos tem Carlos?]
P: O que representa o quatro que você colocou no círculo?
S: Ele significa a diferença de idade entre Carlos e Ricardo.

Invariante 13: O referido em um problema de comparação é obtido a partir da soma do referente com o número que expressa a relação existente entre o referente e o referido.

Este invariante está relacionado com a percepção por parte do sujeito de que o referido em um problema de comparação pode ser obtido pela soma do valor do referente com a relação que existe entre as duas medidas sendo comparadas.

Esse invariante foi observado no discurso de S1, S3 e S4. Um exemplo pode ser visto no discurso de S1 como visto a seguir (Quadro 5.22).

Quadro 5.22 – Fragmento 16 do teste.

FRAGMENTO 16 : PROBLEMA 4 : SUJEITO 1
[Ricardo tem 6 anos, Carlos tem 4 anos a mais do que ele. Quantos anos tem Carlos?]
P: Você colocou o número 10 ao quadrado de cima (referido). Como você explicaria essa ação?
S: Bom, esse é o resultado que eu queria obter, a idade de Carlos. Ele foi obtido a partir da idade de Ricardo somado a diferença de idade entre os dois.

5.3.2 Análise da integração entre as estratégias de scaffolding e a resolução de problemas

Nesta categoria investigamos a contribuição do *scaffolding* no processo de resolução dos problemas de composição, comparação e transformação. Como foi visto, o *scaffolding* disponibilizado no protótipo envolvia:

- Automatização de tarefas;
- Ligação entre as várias representações;
- Uso de representação envolvendo material concreto; e
- *Feedback* para o usuário.

Cada uma dessas funções será vista a seguir.

5.3.2.1 Automatizar tarefas

A construção automática do modelo do diagrama era realizada no momento da seleção do tipo de problema, estando, portanto, disponível para todos os usuários. A ajuda destinada a facilitar o entendimento do significado de cada elemento, entretanto, só seria disponibilizado para aqueles usuários que não conseguissem, após algumas tentativas, realizar a tarefa.

Não foi detectado nenhum problema no uso da construção automática do diagrama. Como era esperado, uma vez exibido na tela o diagrama já montado como

resultado da seleção da estrutura do problema, os usuários passaram a preencher os valores dos elementos logo em seguida. Embora não tenham sido realizados testes nos quais a ausência desse recurso pudesse nos fornecer subsídios que permitissem uma avaliação da influência deste *scaffolding* no processo de resolução, não temos razões para duvidar de sua importância para o aprendiz iniciante.

Em relação ao *scaffolding* relacionado à compreensão do significado dos elementos do diagrama, dados envolvendo a utilização do mesmo bem como da necessidade de oferecer este tipo de auxílio durante a resolução dos problemas surgiram durante os testes.

Durante a resolução dos problemas, ocorreram situações nas quais os sujeitos haviam interpretado corretamente o tipo de problema que lhe foi apresentado, bem como o significado de cada informação fornecida no enunciado. Apesar disso, eles tiveram dificuldades em resolver o problema utilizando a legenda de Vergnaud. Isso pode ser visto no seguinte fragmento extraído do teste com S2 (Quadro 5.23):

Quadro 5.23 – Fragmento 17 do teste.

FRAGMENTO 17 : PROBLEMA 4 : SUJEITO 2
[Ricardo tem 6 anos, Carlos tem 4 anos a mais do que ele. Quantos anos tem Carlos?]
P: Você arrastou o número 6 (o referente) para o círculo (número relativo). Como você explicaria isso? O que o 6 representa para você?
S: O 6 é a idade de Ricardo. Eu coloquei no círculo porque para resolver o problema eu iria somar a idade de Ricardo com a quantidade de anos que Carlos tinha a mais.
P: Você saberia dizer o que deve ser colocado no círculo e nos quadrados em um problema de comparação?
S: Não sei.

Apesar de estar ciente do que era necessário para chegar à solução do problema, o desconhecimento do significado de cada elemento no diagrama impediu S2 de montar o diagrama corretamente. Apenas após utilizar a ajuda que explicava o significado de cada elemento do diagrama, S2 conseguiu resolver este problema. Ao ser solicitado a comentar sobre a ajuda, o sujeito relatou o seguinte (Quadro 5.24):

Quadro 5.24 – Fragmento 18 do teste.

FRAGMENTO 18 : PROBLEMA 4 : SUJEITO 2
[Ricardo tem 6 anos, Carlos tem 4 anos a mais do que ele. Quantos anos tem Carlos?]
P: Após utilizar a ajuda que colocava nomes explicando cada elemento da legenda, você preencheu corretamente os valores do problema. Como você explicaria isso?
S: A ajuda com os nomes mostrou aonde eu deveria colocar cada número, então ficou mais fácil resolver o problema.

O discurso do usuário sugere que a técnica utilizando textos explicativos funciona, entretanto considerações poderiam ser feitas sobre até que ponto o texto deve ajudar o usuário. Em nossa opinião, a escolha do texto de ajuda a ser utilizado deve ser realizada com cuidado, servindo apenas para explicar a função de cada elemento na legenda. A ajuda não deve tirar do usuário a tarefa de interpretar o significado de cada informação disponibilizada no problema.

5.3.2.2 Ligação entre as representações

A ajuda relacionada à ligação entre as representações tinha como objetivo simplificar a interação do usuário com a interface bem como facilitar a percepção por parte do mesmo de que diferentes representações são meios eficazes de fornecer perspectivas diferentes de um mesmo conceito.

Todos os usuários conseguiram utilizar o mecanismo de arrastar números do enunciado para o diagrama. Apesar disso, a facilidade de levar os números direto para o diagrama levou os usuários a não informar o sinal quando o número em questão fosse um número relativo. Isso pode ser visto no discurso de S1 (Quadro 5.25):

Quadro 5.25 – Fragmento 19 do teste.

FRAGMENTO 19 : PROBLEMA 2 : SUJEITO 1
<p>[Ricardo saiu de casa para jogar bola de gude. Ao sair de casa ele possuía 6 bolas. Após o jogo ele tinha duas bolas. O que aconteceu no jogo?]</p> <p>C: Entre agora com o resultado.</p> <p>S: Quatro (apontou este valor para o número relativo).</p> <p>C: Tem certeza que ele ganhou 4 bolas ? Use o sinal de menos ou de mais para representar perda ou ganho. Clique em ok para tentar novamente.</p> <p>S: Ok (Clicou em Ok).</p> <p>S: Menos 4 (apontou este valor para o número relativo).</p> <p>P: Você levou o número quatro ao círculo. Porque você fez isso?</p> <p>S: É exatamente a idéia de que 4 seria a perda, né?</p> <p>P: Você sabia que 4 era a perda, mas não usou o sinal negativo. Porque não usou o sinal de menos?</p> <p>S: Porque não pensei que fosse preciso colocar o sinal.</p>

Esse resultado mostrou a necessidade do software perguntar ao usuário o sinal do número ou trazer o sinal automaticamente no momento em que o número fosse manipulado pelo usuário. A luz dos resultados, optamos por fornecer o sinal para o usuário iniciante, e a medida que o conhecimento do mesmo evolua, retirar essa ajuda, exigindo que o mesmo forneça essa informação.

5.3.2.3 Uso de representação envolvendo material concreto

A ajuda envolvendo a manipulação do material concreto tinha como objetivo facilitar a percepção dos invariantes envolvidos na resolução do problema bem como auxiliar no cálculo da resposta da questão. Da mesma forma que o auxílio relacionado à compreensão do diagrama, a opção por este *scaffolding* só era possível caso o usuário cometesse erros na modelagem do problema. Neste caso, era oferecida a ajuda envolvendo o material concreto, cabendo ao usuário fazer uso dela ou não.

Apenas os usuários S2 e S5 optaram por fazer uso da ajuda que utilizava material concreto. Mesmo quando tiveram oportunidade, os outros usuários preferiram não utilizar este recurso. No caso de S2 e S5, mesmo após o uso desta ajuda, os usuários continuaram errando a modelagem do problema utilizando o diagrama. Solicitado a comentar sobre este fato, S5 mencionou (Quadro 5.26):

Quadro 5.26 – Fragmento 20 do teste.

FRAGMENTO 20 : PROBLEMA 3 : SUJEITO 5
[Maria comprou uma caixa de bombons por 4 reais e ainda ficou com 4 reais. Quanto ela possuía antes de fazer a compra?]
P: Após utilizar a ajuda que envolvia fabricar quadradinhos, você colocou novamente os valores nos mesmos lugares de antes. Porque você fez isso?
S: Porque eu continuei sem saber onde colocar os valores do problema.

Acreditamos que o fracasso da ajuda em auxiliar os usuários na resolução do problema esteja relacionada à ausência de integração entre esta, o problema sendo veiculado e as outras representações disponíveis. Essa questão será tratada na seção 5.4.

5.3.2.4 *Feedback* para o usuário

A criação de mensagens que visam fornecer *feedback* sobre as ações do usuário é uma das partes que mais se beneficia de testes de usabilidade. É difícil avaliar no momento da construção de um protótipo como uma mensagem será interpretada pelo usuário da aplicação. Por esta razão, especial atenção deve ser dada na análise da influência deste tipo de *feedback* nas ações dos usuários.

O papel do *feedback* para o usuário em um software educacional não deve ser subestimado. Um *feedback* bem implementado pode contribuir de forma decisiva no aprendizado dos conceitos veiculados na interface. Neste sentido, o teste realizado forneceu dados importantes para a melhoria do *feedback* fornecido pelo sistema.

Os dados obtidos nos testes relacionados ao *feedback* fornecido ao usuário foram analisados em relação a:

- Adequação da mensagem às necessidades do usuário; e

- Interpretação do erro cometido pelo usuário.

Nesta seção comentaremos apenas trechos do teste nos quais o *feedback* fornecido pelo sistema se mostrou insuficiente ou inadequado para atender às necessidades dos usuários. Acreditamos que os fragmentos mostrados revelam aspectos importantes para o projeto do sistema de ajuda em um software educacional.

Adequação da mensagem às necessidades do usuário

É importante que o *feedback* fornecido em um software educacional seja claro e ajude o usuário a entender a razão do erro cometido. Não basta apenas informar que a ação que o indivíduo realizou está errada. Além disso, a lógica de funcionamento das mensagens deve ser consistente, para facilitar a adaptação do usuário ao software.

Um outro ponto importante é estar certo de que toda a informação necessária ao entendimento do erro cometido pelo usuário esteja disponível ao mesmo no *feedback* fornecido através da mensagem. Acesso a informações adicionais que sirvam para tirar dúvidas relacionadas ao erro cometido também deve ser oferecido neste tipo de ajuda.

A necessidade do sistema fornecer informações adicionais que ajudassem o usuário a compreender o erro cometido apareceu em algumas ocasiões durante o teste. Essa situação pode ser vista a seguir nos fragmentos do discurso de S2 e S3 (Quadros 5.27 e 5.28):

Quadro 5.27 – Fragmento 21 do teste.

FRAGMENTO 21 : PROBLEMA 4 : SUJEITO 2
[Ricardo tem 6 anos, Carlos tem 4 anos a mais do que ele. Quantos anos tem Carlos?]
P: Após receber a mensagem do computador “Tem certeza que o referente está na posição correta?”, você pediu mais uma dica. A mensagem não esclareceu o que estava errado no problema?
S: Ela disse que eu errei, mas não ajudou a entender o que houve.

Quadro 5.28 – Fragmento 22 do teste.

FRAGMENTO 22 : PROBLEMA 4 : SUJEITO 3
[Ricardo tem 6 anos, Carlos tem 4 anos a mais do que ele. Quantos anos tem Carlos?]
P: Após receber a mensagem do computador “Tem certeza que é uma junção entre duas quantidades?”, você continuou optando pela composição. Como você explicaria isso?
S: Porque no problema eu estou juntando a idade de Ricardo com os 4 anos que Carlos tem a mais. Então seria uma junção de duas quantidades. A mensagem então reforçou minha opção pela composição.
P: A mensagem então não o ajudou a entender o erro?
S: Ela apenas me disse que estava errado, mas não dizia porque. Ela poderia explicar o significado da composição, para que a pessoa entendesse o que está errado.

O fornecimento de *feedback* insuficiente pelo software educacional também pode acarretar no uso, por parte do usuário, de estratégias inadequadas para a resolução do problema que comprometerão a aprendizagem do conteúdo trabalhado na interface. No fragmento 23, a ausência de explicações adicionais que pudessem esclarecer a dúvida do usuário acabou levando-o a escolher a estrutura adequada para o problema veiculado na tela por eliminação. Essa situação pode ser vista no discurso de S3 a seguir (Quadro 5.29):

Quadro 5.29 – Fragmento 23 do teste.

FRAGMENTO 23 : PROBLEMA 3 : SUJEITO 3
[Maria comprou uma caixa de bombons por 4 reais e ainda ficou com 4 reais. Quanto ela possuía antes de fazer a compra?]
P: Após receber a mensagem do computador “Tem certeza que é uma junção entre duas quantidades?”, você optou pela transformação. Como você explicaria essa decisão?
S: Bom, na transformação, da mesma forma que na composição, você também está operando com duas quantidades. Se eu tenho 4 bolas e ganhei 3 bolas, também se trata de uma composição. Então os problemas de composição e de transformação são muito parecidos. Quando eu vi a mensagem, percebi que não era composição. Então como eu sei que não estou comparando nada neste problema, só pode ser transformação.

Neste fragmento, o usuário inicialmente optou pela composição como o tipo de problema sendo veiculado pela interface. A mensagem não o ajudou a entender a diferença entre composição e transformação, apenas alertando-o de que a escolha pela composição era equivocada. Ao relatar que sua opção pela transformação se deu por eliminação, o usuário deixa claro o fracasso da mensagem em fornecer informações que permitissem ao mesmo entender o erro cometido.

Durante o projeto do *feedback* a ser dado pela interface, algumas coisas passaram despercebidas e foram notadas pelos usuários. S5 reclamou do *feedback* dado pelo software, que achou incompleto (Quadro 5.30).

Quadro 5.30 – Fragmento 24 do teste.

FRAGMENTO 24 : SUJEITO 5
S: O computador avisa quando o valor está no lugar errado, mas não avisou quando o valor estava no lugar certo. Poderia ter dito “muito bem”, “pode continuar”, falado algo que mostrasse que eu acertei.

Consideramos que adequar o *feedback* às necessidades dos usuários é muito importante no projeto de um software educacional. Na seção 5.4 comentaremos algumas modificações realizadas no sistema de *feedback* da aplicação de forma a torná-lo mais adequado aos seus usuários.

Interpretação do erro cometido pelo usuário

Interpretar as razões que levaram o usuário a cometer um erro em um software educacional de forma a fornecer um *feedback* adequado não é uma tarefa simples de resolver. Diversos motivos podem levar um aprendiz a cometer um erro durante o uso de um software educacional. Identificar o conhecimento do usuário sobre o assunto veiculado na interface bem como a experiência do mesmo com o software pode ser um bom caminho para se fornecer uma ajuda adequada às necessidades de cada usuário em particular.

A análise dos erros cometidos pelo usuário poderia se beneficiar de um modelo sobre o aprendiz, que permitisse avaliar, a partir da inferência do conhecimento do usuário sobre o assunto tratado, qual a explicação mais provável para um determinado erro, permitindo ao software fornecer *feedback* personalizado. A inferência do conhecimento do usuário poderia ser realizada a partir de dados armazenados sobre as ações passadas realizadas pelo mesmo com o software. À medida que o aprendiz utilizasse o sistema, este iria aprendendo cada vez mais sobre o primeiro, e poderia utilizar esse conhecimento para fornecer *feedback* adequado.

Na ausência de um modelo de cada um dos indivíduos que participariam do teste, a decisão sobre a interpretação dos erros que poderiam ser cometidos pelos usuários durante o uso do protótipo foi baseada em nossa intuição sobre os mesmos. Consideramos que seria mais provável o usuário errar devido a uma falha na interpretação do problema, e que erros causados pela falha no entendimento do diagrama seriam menos frequentes. As mensagens utilizadas no protótipo utilizaram este raciocínio, direcionando o foco em auxiliá-lo na compreensão do problema, em vez de focar no entendimento do diagrama.

Uma escolha realizada desta forma, sem dados que sustentem a decisão tomada, pode produzir alguns problemas. Uma mensagem que interprete o erro cometido pelo

usuário de forma errada pode confundi-lo, levando-o a cometer novos erros. Essa situação pode ser vista no fragmento 25 (Quadro 5.31):

Quadro 5.31 – Fragmento 25 do teste.

FRAGMENTO 25 : PROBLEMA 2 : SUJEITO 1
[Ricardo saiu de casa para jogar bola de gude. Ao sair de casa ele possuía 6 bolas. Após o jogo ele tinha duas bolas. O que aconteceu no jogo?]
S: dois (Arrastou o número 2 do texto do problema (estado final) e posicionou no círculo (número relativo)).
C: Tem certeza que o valor do número relativo está correto ? O número relativo é apresentado da seguinte forma: +2 ou -2. Clique em ok para tentar novamente.
S: menos dois (apontou este valor para o círculo).
P: Você arrastou o número 2 para o círculo (o número relativo). Porque você levou o número dois ao círculo?
S: Porque o dois é o estágio final do problema, é o resultado. É a quantidade de bolas que resta a Ricardo.
P: Após receber a mensagem do computador, você optou por colocar o valor 2 no círculo novamente, desta vez com sinal negativo. Porque você levou o valor menos dois ao círculo?
S: Pela informação dada pelo computador. Achei que o problema fosse o sinal.

Como pode ser visto a partir da explicação do usuário, este havia interpretado corretamente o significado do número 2 no problema, mas associou este valor ao elemento errado do diagrama, neste caso o círculo. A mensagem partiu do pressuposto que o usuário entendia o que o círculo representa em um diagrama de transformação, e que o erro do indivíduo seria relacionado ao entendimento do significado do número 2 na questão. Esta falha na interpretação do erro do usuário não apenas impediu o mesmo de compreender qual tinha sido o seu engano, mas ainda levou-o a cometer um novo erro.

A estratégia de analisar o erro deste usuário considerando que houve uma falha na utilização do diagrama se mostrou mais acertada. O fragmento 26, referente ao mesmo usuário, sugere isso (Quadro 5.32).

Quadro 5.32 – Fragmento 26 do teste.

FRAGMENTO 26 : PROBLEMA 4 : SUJEITO 1
[Ricardo tem 6 anos, Carlos tem 4 anos a mais do que ele. Quantos anos tem Carlos?]
S: Seis... (Arrastou o número 6 do texto do problema (o referente) e posicionou no quadrado referido).
C: Tem certeza que o referente está na posição correta? Clique em Ok para tentar novamente.
S: Seis... (Arrastou o número 6 do texto do problema (o referente) e posicionou no quadrado referente).
P: Você arrastou o número 6 para este quadrado (o referido). Porque você fez isso?
S: Bom, não me pareceu muito claro o local onde a idade de Carlos deveria ser colocada. Por isso coloquei aí.
P: E qual era a sua intenção ao realizar esta ação? O que o número 6 representava para você?
S: O 6 representava a idade a qual deveria ser adicionada uma quantidade de anos para obter a idade do outro sujeito. O 6 então seria o ponto de partida para a solução do problema, seria o referencial.

De uma forma geral, foi possível coletar durante o teste informações importantes sobre as necessidades de *feedback* dos usuários. Essas informações foram utilizadas para aperfeiçoar o sistema de ajuda do protótipo. Mais detalhes sobre isso serão vistos na seção a seguir.

5.3.3 Discussão dos resultados dos testes de usabilidade

A análise dos dados obtidos no teste revelou aspectos importantes do protótipo desenvolvido neste trabalho.

Em relação aos invariantes mobilizados, observamos uma quantidade significativa de propriedades de conceito emergir no discurso dos usuários quando os mesmos eram solicitados a justificar as ações realizadas durante a resolução de problemas com o protótipo. A presença de invariantes nas explicações fornecidas pelos usuários para as ações realizadas com o protótipo nos forneceu evidências da capacidade da interface de promover a emergência de significados relacionados ao campo conceitual aditivo. Podemos então concluir que, em relação a esse aspecto, a aplicação atingiu os objetivos que motivaram a sua construção.

Os testes também tinham como objetivo investigar a contribuição do *scaffolding* disponível na interface ao processo de resolução de problemas. Neste sentido, a avaliação mostrou que partes do *scaffolding* cumpriram seu papel adequadamente, enquanto outras necessitam de correções para atingir todo o seu potencial. Essas correções foram realizadas à luz dos dados coletados, os quais forneceram informações valiosas sobre as necessidades dos usuários e como estas podem ser atendidas.

A automatização de tarefas bem como o uso de textos explicativos para os elementos da legenda foram bem recebidos pelos usuários, que conseguiram fazer uso desses recursos durante o uso da interface. Os resultados obtidos sugerem que estratégias nesse sentido são úteis para os usuários do software educacional, devendo ser utilizadas sempre que possível.

A ligação entre as representações foi outro recurso que funcionou de maneira satisfatória. Este recurso procura facilitar o aprendizado dos conceitos na medida em que destaca a conexão existente entre diferentes representações, que deixam de ser vistas como formas isoladas de comunicar as propriedades dos conceitos. Como foi visto no teste, cuidado extra deve ser tomado ao projetar a conexão entre as representações. A facilidade de passar informações de uma representação para outra pode encobrir aspectos importantes para a aprendizagem dos conceitos por parte dos usuários. A avaliação da integração entre as representações durante o teste com os usuários é uma forma eficaz de prevenir este problema.

Se por um lado as duas primeiras estratégias de *scaffolding* funcionaram de acordo com o previsto, o mesmo não pode ser dito em relação ao uso do material concreto. Para entendermos o resultado que emergiu dos testes, devemos considerar inicialmente porque esta ajuda foi projetada da forma vista neste capítulo.

A representação envolvendo o uso do material concreto foi criada com o objetivo de viabilizar o uso no software de um recurso utilizado em sala de aula como apoio à aprendizagem. O projeto deste recurso também tinha como meta disponibilizar uma ajuda genérica para cada tipo de situação de forma a facilitar a criação de novos problemas para uso no software. Desta forma, a ajuda destinada a questões envolvendo parte-todo poderia ser utilizada em problemas similares que também envolvessem composição, o mesmo valendo para a ajuda de transformação e comparação.

Acreditávamos que os usuários não teriam problemas em relacionar a metáfora presente no uso do material concreto com o problema sendo resolvido. Isto não aconteceu. Os resultados indicaram que o usuário pode ter dificuldades em relacionar o raciocínio utilizado em uma ajuda disponibilizada em um contexto diferente do problema que está sendo apresentado pelo software.

Outro ponto a ser destacado sobre a ajuda é que ela foi feita apenas para auxiliar na percepção de invariantes bem como no cálculo da solução do problema. Não foi considerada a possibilidade da mesma colaborar também no entendimento do funcionamento do diagrama. Os testes mostraram que o uso do diagrama provoca muitas dúvidas nos usuários iniciantes, sendo necessário esclarecer mais o funcionamento da legenda para o usuário. A nova versão da ajuda envolvendo material concreto procura dar sua contribuição para solucionar esse problema, através do uso de uma ligação entre essas duas representações. As modificações realizadas neste *scaffolding* serão abordadas com mais detalhes na seção seguinte.

A última estratégia de *scaffolding* presente no protótipo tinha como meta fornecer *feedback* para o usuário através do uso de mensagens. Os testes mostraram que algumas mensagens cumpriram o seu objetivo de elucidar dúvidas sobre o erro cometido pelo usuário enquanto outras não foram bem sucedidas neste aspecto. As justificativas fornecidas pelos usuários sobre as ações realizadas após o aparecimento de tais mensagens indicam que a falta de informações adicionais sobre o assunto relacionado ao erro foi decisivo em algumas situações para o insucesso verificado. Outro aspecto destacado pelos usuários durante os testes foi quanto a ausência de *feedback* para as ações corretas. Essas demandas foram levadas em consideração no design da versão final do software, que está agora em processo de adaptação para integração ao ambiente Amadeus.

5.4 Implementação do software

Uma vez que o protótipo tenha sido validado nos testes realizados com usuários representativos, a implementação do software pode começar. Durante os testes, algumas partes do sistema de ajuda não funcionaram como deveriam. É necessário então propor uma solução para corrigir essas deficiências no software a ser construído.

Neste sentido, as modificações realizadas no sistema de ajuda tiveram como objetivo corrigir as deficiências observadas no teste em relação ao uso do material concreto bem como no *feedback* fornecido pelo usuário.

Em relação ao *feedback* fornecido pelo sistema, os usuários reclamaram da ausência do mesmo para as ações positivas bem como da necessidade de informações adicionais para o entendimento, por parte do usuário, dos erros cometidos na modelagem dos problemas. Enquanto o primeiro problema pode ser resolvido adicionando *feedback* a cada ação correta realizada, a resolução do segundo problema passa pela difícil questão de saber quanto de informação adicional deve ser inserido no *feedback* de forma a esclarecer a dúvida do usuário.

Uma hipótese razoável é que a quantidade de informação necessária a esclarecer uma dúvida do usuário dependerá do conhecimento deste indivíduo sobre o assunto tratado bem como da eficiência do software em explicar o problema ocorrido. Uma consequência disso é que apenas o usuário é capaz de informar quando sua dúvida foi esclarecida. Desta forma, cabe ao usuário indicar ao software a quantidade adicional de informação que o mesmo necessita.

Esta análise nos leva a propor que o sistema de ajuda seja ampliado com informações sobre as situações que fazem parte do campo conceitual aditivo e que esta ajuda seja interligada ao sistema de mensagens do software. Desta forma, o usuário pode passar de uma mensagem direto para informações relacionadas ao erro cometido por ele. Uma forma de viabilizar isso é adicionar um botão “mais informações” em cada mensagem do software, vinculando esse botão a uma explicação específica (Figura 5.25). Caso a informação adicional não seja suficiente, o usuário poderá seguir solicitando informações adicionais até que esteja satisfeito.

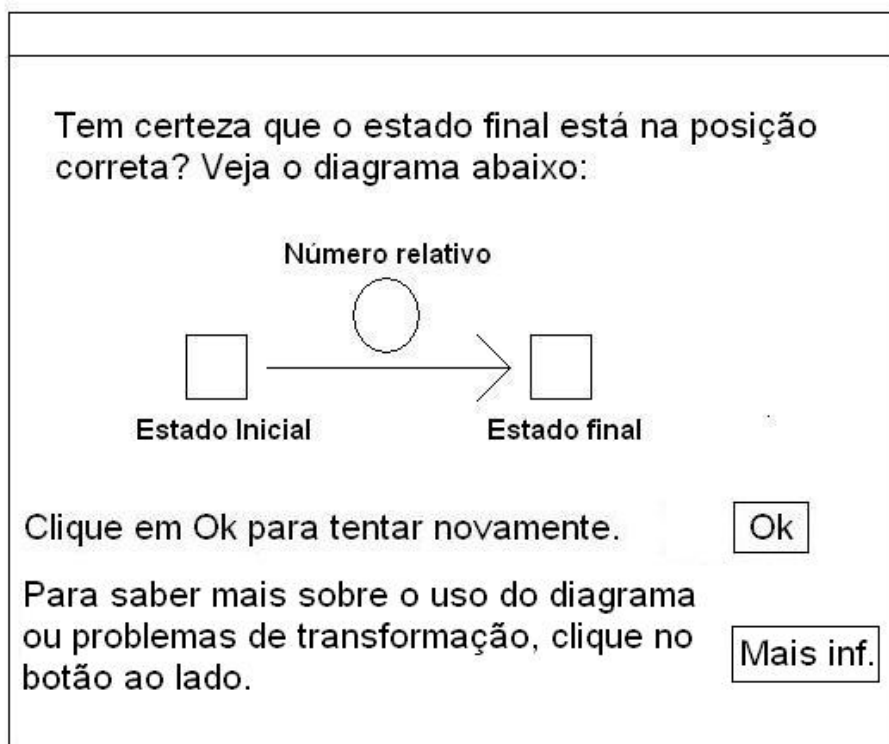


Figura 5.25 – Mensagem com link para informações adicionais.

Em relação ao uso do material concreto, a dificuldade do usuário em relacionar a situação apresentada em cada uma das ajudas disponibilizadas com a questão veiculada pelo software nos indica a necessidade dessa ajuda refletir fielmente o problema sendo resolvido pelo usuário. A nova ajuda envolvendo o uso do material concreto levará o indivíduo a manipular representações físicas dos elementos presentes no problema dentro do mesmo contexto existente na questão. Um exemplo (Figura 5.26) pode ser visto para o problema 1 utilizado em nosso teste, repetido a seguir por conveniência (Quadro 5.33).

Quadro 5.33 – Problema 1

Ao redor da mesa estão sentados 4 garotos e 7 garotas. Quantas pessoas estão sentadas ao redor da mesa?

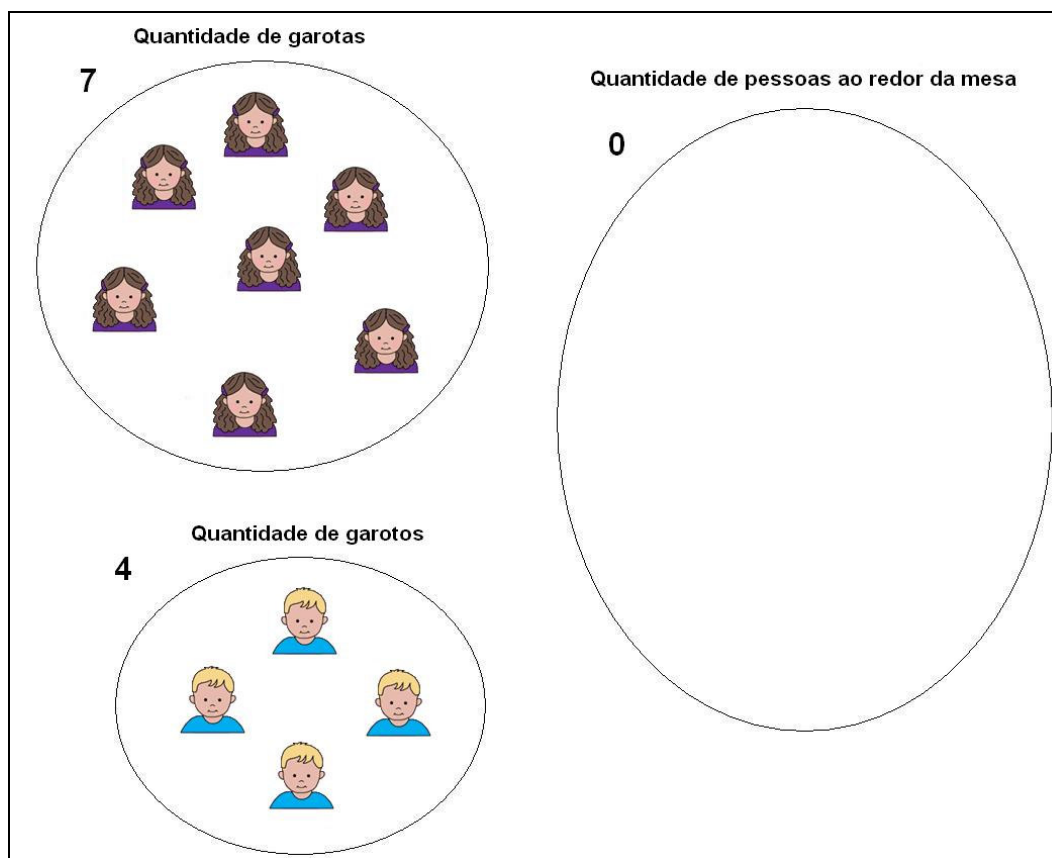


Figura 5.26 – Ajuda para a composição modificada.

A modificação implementada na ajuda não se resume apenas à manipulação dos próprios elementos dentro do mesmo contexto tratado no problema. Para reforçar a ligação entre a atividade com o material concreto e o problema sendo resolvido pelo usuário, decidimos permitir que os elementos do material concreto possam ser levados para o diagrama. Desta forma, após realizar a atividade com o material concreto, o usuário pode preencher os valores do diagrama simplesmente arrastando os elementos direto da ajuda com material concreto.

O Gérard (Figura 5.27) está sendo implementado na linguagem Java, de forma a ser integrado no ambiente virtual de ensino AMADeUs. O AMADeUs (Agentes Micromundo e A.D.e.C.U.I.) está sendo desenvolvido pelo grupo de Ciências Cognitivas e Tecnologia Educacional (CCTE), ligado ao Centro de Informática (CIn) da Universidade Federal de Pernambuco (UFPE). Este projeto tem como objetivo disponibilizar um ambiente virtual de aprendizagem baseado no conceito de micromundos, bem como ferramentas e metodologias que viabilizem a realização de cursos a distância.

A linguagem Java foi escolhida para implementação do Gérard pelo fato de ser um ambiente multiplataforma, o que permitirá o uso do Gérard em diferentes plataformas computacionais, viabilizando a sua adaptação aos diferentes sistemas existentes em escolas e na residência dos usuários.

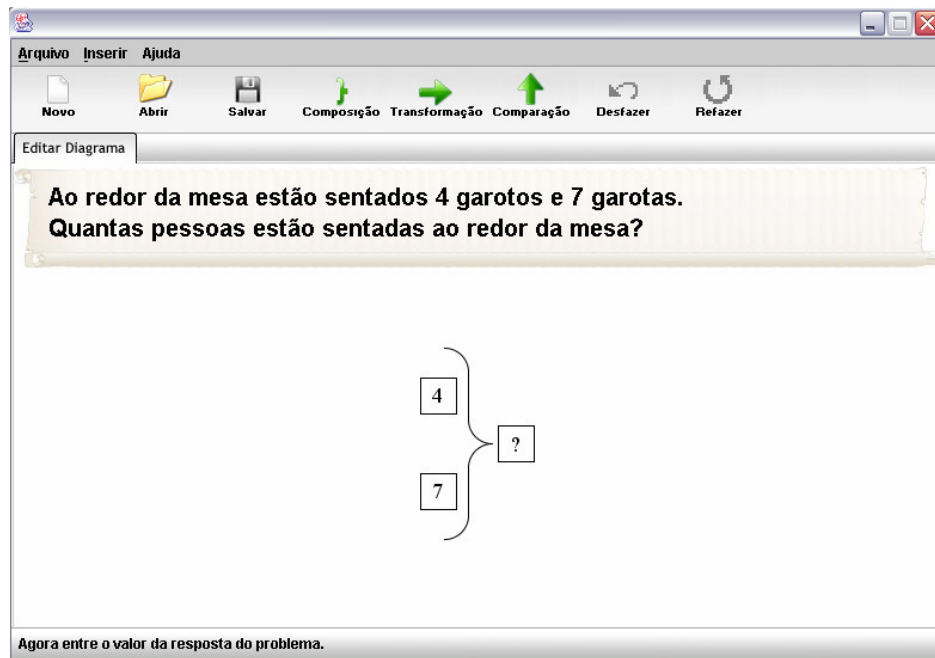


Figura 5.27 – Software Gerard.

6 Conclusões

Conforme mencionado no capítulo 4, o principal objetivo deste trabalho era projetar um software educacional a partir de um modelo teórico que identificasse os requisitos para aprendizagem de um determinado domínio, de forma a viabilizar uma avaliação do conhecimento mobilizado por usuários durante o uso da aplicação.

A busca por um referencial teórico que preenchesse esses requisitos nos levou a conhecer as pesquisas desenvolvidas por Gérard Vergnaud no campo da educação matemática. A teoria dos campos conceituais proposta por este pesquisador fornece um ferramental teórico adequado para conceber e avaliar interfaces educacionais para o ensino de matemática. O conhecimento desta teoria bem como do suporte à aprendizagem através do *scaffolding* permitiu identificar as etapas necessárias ao desenvolvimento de software educacional baseado nesses dois paradigmas.

A necessidade de dispor de uma versão utilizável do protótipo o mais cedo possível para a realização dos testes com o usuário nos levou a desenvolver um protótipo de baixa fidelidade do software a ser desenvolvido. A prototipação é uma técnica eficaz para testar hipóteses e conhecer as necessidades dos usuários ainda no início do desenvolvimento, quando realizar alterações custa mais barato. Em nosso trabalho, ela permitiu iniciar o desenvolvimento do software apenas após os requisitos terem sido validados em uma avaliação prática com usuários representativos, o que aumentou a nossa confiança de que a aplicação poderá cumprir os objetivos que motivaram a sua construção.

Desenvolver o protótipo exigiu também o levantamento dos requisitos de uma aplicação educacional que não eram ligados diretamente à aprendizagem do domínio veiculado na interface ou às necessidades dos alunos. Este levantamento foi realizado a partir da análise de soluções concorrentes. A análise dos competidores se revelou uma técnica útil que forneceu os requisitos restantes para a implementação do protótipo.

O método de avaliação empregado neste trabalho reflete a nossa visão de que a qualidade de um software educacional deve ser medida em função da aprendizagem que decorre do uso desse software educacional. Até onde nós sabemos, esta é uma abordagem original para o desenvolvimento de interfaces educativas.

A teoria dos campos conceituais é um quadro teórico cujo potencial para colaborar no aprendizado da matemática ainda não foi totalmente explorado. Entre suas contribuições para o ensino está o entendimento de que os requisitos para a aprendizagem bem como as dificuldades enfrentadas pelos alunos no aprendizado de um determinado campo conceitual são inerentes a este domínio em particular. Esperamos que este trabalho ajude a difundir o uso desta teoria na concepção e avaliação de software educacional para o ensino de matemática.

Veremos a seguir as principais contribuições de nosso trabalho, bem como as dificuldades encontradas durante a realização de nossas pesquisas. Por fim, concluiremos este capítulo com indicações de trabalhos futuros.

6.1 Contribuições

A principal contribuição deste trabalho é o software desenvolvido para ensino das estruturas aditivas. Este software, que recebeu o nome de Gérard em homenagem ao pesquisador que desenvolveu o quadro teórico que guiou o seu desenvolvimento, é uma aplicação que foi concebida de forma a mobilizar os invariantes necessários a aprendizagem do campo conceitual das estruturas aditivas. Seu estágio atual nos dá confiança de que esta interface educativa pode cumprir os objetivos que motivaram o seu desenvolvimento.

Partindo do pressuposto de que o valor de uma aplicação educacional deve ser medido a partir da aprendizagem que emerge a partir do seu uso, este trabalho pretende contribuir também demonstrando como esta avaliação pode ser realizada. Para isto nos baseamos em quadros teóricos desenvolvidos no âmbito da psicologia cognitiva que nos permitiram inferir o conhecimento mobilizado por um usuário ao utilizar um software educacional concebido para colaborar no aprendizado das estruturas aditivas.

Como outras contribuições deste trabalho, podemos mencionar:

- Chamar a atenção para a necessidade de avaliações centradas na análise da aprendizagem em contexto de uso do software; e
- Definição de etapas para a criação de software educacional baseado na teoria dos campos conceituais e no suporte à aprendizagem através do *scaffold*.

6.2 Dificuldades encontradas

Durante o desenvolvimento deste trabalho, alguns obstáculos tiveram que ser superados. Listamos a seguir os principais problemas com que tivemos de lidar durante a realização desta pesquisa.

6.2.1 Interdisciplinaridade

Em nosso trabalho utilizamos quadros teóricos advindos da psicologia cognitiva. A utilização de teorias provenientes de uma área de conhecimento com a qual não tínhamos até esta ocasião nenhum contato prévio se traduziu na necessidade de buscar inicialmente textos introdutórios que nos habilitassem a iniciar nossos estudos sobre as teorias de aprendizagem de forma a poder utilizá-las em nossa pesquisa.

6.2.2 Professores como usuários

O software desenvolvido neste trabalho tem como objetivo ser utilizado em cursos de capacitação de professores a distância. A baixa remuneração oferecida aos profissionais da educação no Brasil, sobretudo aos que trabalham no ensino fundamental, levam os professores a assumir uma quantidade de horas em sala de aula incompatível com outras necessidades da atividade docente, como participar de cursos de qualificação e aperfeiçoamento bem como se inteirar sobre pesquisas relacionadas à prática de ensino. Além disso, participar de uma pesquisa que envolvia resolver problemas matemáticos utilizando um software poderia parecer arriscado do ponto de vista do docente, que se sentiria constrangido caso não conseguisse realizar as tarefas propostas. Como resultado disso, tivemos muita dificuldade em selecionar professores para participar de nossa pesquisa, pois os docentes não dispunham de tempo ou incentivo para participar dos testes de nossa aplicação. Isso levou a atrasos em nossa pesquisa, pois foi necessário que nos adaptássemos aos locais, datas e horários estipulados pelos professores como condição para participar de nossa pesquisa.

6.3 Trabalhos futuros

O software desenvolvido neste trabalho forneceu resultados satisfatórios em relação aos objetivos que motivaram a sua construção. Consideramos, entretanto, que há espaço para adição de melhorias que, devido à limitação de tempo, não puderam ser implementadas. Essas melhorias são mencionadas a seguir.

Novos usuários

O software Gérard foi projetado para ser utilizado em cursos de capacitação para professores que lecionam no primeiro grau. Uma extensão natural desta aplicação seria adequar o mesmo para ser utilizado também por crianças que cursam o ensino fundamental.

Uma ampliação do público alvo do Gerard exigiria mudanças no software de forma a contemplar as necessidades desses novos usuários. No caso específico de crianças, uma nova representação para os conceitos, mais adequada a esses aprendizes, poderia ser desenvolvida e adicionada ao software. O uso de elementos lúdicos poderia também ser utilizado como forma de aumentar a motivação no uso desta interface educativa.

Modelo do usuário

A estratégia de ensino baseada em *scaffold* requer que o suporte à aprendizagem oferecido em uma interface educacional desapareça gradualmente, à medida que o usuário evolui. O desaparecimento da ajuda aumenta o grau de liberdade do usuário na utilização da interface bem como na realização da tarefa.

Em nosso software, o nível de ajuda é determinado pela escolha prévia do perfil do usuário (iniciante ou experiente) antes de iniciar o uso da aplicação. A mudança de nível de iniciante para experiente também é realizada durante o uso do sistema, e utiliza como critério o número de problemas resolvidos corretamente por parte do usuário.

Acreditamos que há espaço para utilização de um modelo mais preciso para avaliação do conhecimento atual do aprendiz sobre o domínio veiculado na interface. Isso poderia ser feito através da inferência do conhecimento do usuário a partir das ações passadas realizadas pelo mesmo durante o uso do sistema. Desta forma, a cada utilização do software novas informações seriam agregadas à base de informações armazenadas sobre o usuário, permitindo ao sistema a retirada gradual da ajuda a partir da evolução do aprendiz.

A avaliação do progresso do usuário no aprendizado do conteúdo veiculado por um software educacional está sendo tratado por uma colega de mestrado, Ana Emília de Melo Queiroz, em seu trabalho de pesquisa [QUE06].

7 Referências


- [ATA03] Atayde A., Teixeira A., Pádua C. MAQSEI - uma Metodologia de Avaliação de Qualidade de Software Educacional Infantil. - XIV Simpósio Brasileiro de Informática na Educação - SBIE - NCE/UFRJ 2003.
- [BRA03] Braga M., Gomes A., Análise crítica do contexto de jogo em software educativo. 2o Workshop de Jogos e Entretenimento Digital, WJogos 2003, Salvador, 2003.
- [BUR99] Burd L. Desenvolvimento de software para atividades Educacionais. Dissertação de Mestrado, Faculdade de Engenharia Elétrica, Unicamp, 1999.
- [CAL06] Caltrox Educational Software. Math Logic. Disponível em: <http://www.caltrox.com/products/mathlogic.htm> (07/02/06).
- [CAM94] Campos G. Metodologia para avaliação da qualidade de software educacional: Diretrizes para desenvolvedores e usuários. Tese de doutorado, Universidade Federal do Rio de Janeiro, 1994.
- [CAR90] Carraher D. O que esperamos de um software educacional?. Acesso: Revista de Educação e Informação, Ano II, n. 3, jan/fev, 1990.
- [CHA03] Chaves E. O que é software educacional? Disponível em http://www.jlcosta.passos.com.br/complementa/Utilizacao_do_computador.doc (05/08/03).
- [DAV06] Davidson & Associates. Math Blaster 2: The Secret of the Lost City. Disponível em: <http://www.tabone.com.mt/davidson/blast2.htm> (07/02/06).
- [FRA99] Franchi A. Considerações sobre a teoria dos campos conceituais. In Alcântara Machado, S. et al. Educação matemática: uma introdução. São Paulo. Educ, pp. 155-195, 1999.
- [GAM98] Gamez L. TICESE: Técnica de inspeção de conformidade ergonômica de software educacional. Dissertação de mestrado, Escola de Engenharia, Universidade do Minho, Portugal, 1998.
- [GOM00] Gomes A. Um modelo construtivista para análise da ação com instrumentos: a aprendizagem consecutiva ao uso de artefatos computacionais. Anais do V Congresso Ibero-Americano de Informática Educativa, v.1, 2000.

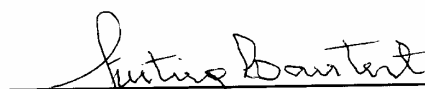
- [GOM02] Gomes A., Castro-Filho J., Gitirana V., Spinillo A., Alves M., Melo M., Ximenes J. Avaliação de software educativo para o ensino de matemática. E. F. Ramos (ed.), *Convergências Tecnológicas – Redesenhando as fronteiras da Ciência e da Educação: Anais*. Florianópolis, SBC 2002.
- [GRE93] Great Wave Software. Kids Math, 1993.
- [HIN01] Hinostroza J., Mellar H. Pedagogy embedded in educational software design: report of a case study, *Computers & Education*, Vol. 37, pp. 27–40, 2001.
- [KNI06] Knightlite Software. Ahsha Math. Disponível em: <http://knightlite.net/math/> (07/02/06).
- [KOT98] Kotonya G., Sommerville I. *Requirements Engineering: Processes and techniques*. Wiley, 1998.
- [MAG01] Maguire M. Methods to support human-centred design. *International Journal of Human-Computer Studies*, Vol. 55, n° 4, pp. 587-634, 2001.
- [MAG02] Magina S., Campos T., Nunes T., Gitirana V. *Repensando Adição e Subtração: contribuições da teoria dos campos conceituais*. 2° Edição. Ed. PROEM, 2002.
- [MCF02] McFarlane A., Sparrowhawk A., Heald Y., Report on the educational use of games: An exploration by TEEM of the contribution which games can make to the education process, 2002. http://www.teem.org.uk/resources/teem_gamesined_full.pdf (02/08/03).
- [MOR02] Moreira M. A teoria dos campos conceituais de Vergnaud, o ensino de ciências e a pesquisa nesta área. *Investigações em Ensino de Ciências*, Porto Alegre, v. 7, n. 1, 2002. Disponível em http://www.if.ufrgs.br/public/ensino/vol7/n1/v7_n1_a1.html (23/11/05).
- [NIE93] Nielsen J. *Usability Engineering*. Academic Press, 1993.
- [NOR86] Norman, D., DRAPER, S. *User centered system design*. Hillsdale, Lawrence Erlbaum Associates, 1986.
- [NUS99] Nussbaum M., Rosas R., Rodríguez P., Sun Y. e Valdivia V. *Diseño, Desarrollo y Evaluación de Video Juegos Portátiles Educativos y Autorregulados*, 1999. Disponível em: <http://www.ciencia.cl/CienciaAIDia/volumen2/numero3/articulos/articulo1.html> (02/08/03).

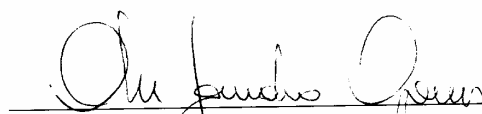
- [PRE02] Preece J., Rogers Y., Sharp H. Interaction Design: Beyond Human-Computer Interaction. John Wiley & Sons, 2002.
- [QSR06] QSR International. Non-Numerical Unstructured Data Indexing Searching and Theorizing (Nud*ist). Disponível em: <http://www.qsr.com.au/software.htm> (07/02/06)
- [QUE06] Queiroz, A. Desenvolvimento de agentes inteligentes a partir da modelagem do usuário com técnicas de IHC. Dissertação de mestrado não publicada, Pós-Graduação em Ciência da Computação, Universidade Federal de Pernambuco, 2006.
- [RIC01] Rick J. AudioExplorer: Multiple Linked Representations for Convergence. GVU-Technical Report GIT-GVU-01-15, 2001.
- [SIL98] Silva C. Bases pedagógicas e ergonômicas para concepção e avaliação de produtos educacionais informatizados, 1998. Disponível em: <http://www.eps.ufsc.br/disserta98/ribeiro/> (05/08/03).
- [SNY03] Snyder C. Paper Prototyping: The Fast and Easy Way to Design and Refine User Interfaces. Morgan Kaufmann, 2003.
- [SOL94] Soloway E., Guzdial M., Hay K. Learner-Centered Design: The Challenge for HCI In The 21st Century. interactions, v.1, n.2, p.36-48, 1994.
- [SPI04] Spinillo A., Magina G. Alguns mitos sobre a educação matemática e suas consequências para o ensino fundamental. In: Matemática nas séries iniciais do ensino fundamental: A pesquisa e a sala de aula. Ed. São Paulo: Biblioteca do Educador Matemático, p. 7-35, 2004.
- [VAL93] Valente J. Computadores e conhecimento: repensando a educação. Universidade Estadual de Campinas – Unicamp, 1993. Disponível em: <http://www.nied.unicamp.br/publicacoes/pub.php?classe=separata> (08/11/05).
- [VAL97] Valente J., Almeida J. Visão Analítica da Informática na Educação no Brasil: a questão da formação do professor. Revista Brasileira de Informática na Educação, Florianópolis, Sociedade Brasileira de Computação, n. 01, p. 45-60, 1997.
- [VER82] Vergnaud, G. A classification of cognitive tasks and operations of thought involved in addition and subtraction problems. Em Carpenter, T., Moser, J. & Romberg, T. Addition and subtraction: A cognitive perspective. Hillsdale, N.J.: Lawrence Erlbaum, p. 39-59, USA, 1982.

- [VER90] Vergnaud G. La Théorie des champs conceptuels. Recherches en didactique mathématique. Grenoble, La Pensée Sauvage, n° 6, vol. 10, n° 2,3, p. 133-170, 1990.
- [VER91] Vergnaud G. El niño, las matemáticas y la realidad. Problemas de la enseñanza de las matemáticas en la escuela primaria. México: Trilhas, 1991.
- [VER97] Vergnaud G. The Nature of Mathematical Concepts. Em Nunes, T. & Bryant, P. (eds) Learning and Teaching Mathematics: An International Perspective, Psychology Press, East Sussex, pp. 5-28, 1997.
- [VIE99] Vieira F. Avaliação de software educativo: Reflexões para uma análise criteriosa. Disponível em: http://www.nuted.edu.ufrgs.br/biblioteca/public_html/9/30/index.html (03/12/05).
- [WOO76] Wood D., Bruner J., Ross G. The role of tutoring in problem solving. Journal of Child Psychology and Psychiatry, 17, pp. 89-100, 1976.

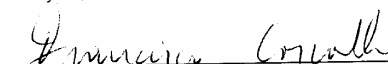
Dissertação de Mestrado apresentada por **Maurício da Motta Braga** à Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título “**Design de Software Educacional Baseado na Teoria dos Campos Conceituais**”, orientada pelo **Prof. Alex Sandro Gomes** e aprovada pela Banca Examinadora formada pelos professores:


Prof. Fernando da Fonseca de Souza
Centro de Informática / UFPE


Profa. Síntria Labres Lautert
Departamento de Psicologia / UFPE


Prof. Alex Sandro Gomes
Centro de Informática / UFPE

Visto e permitida a impressão.
Recife, 6 de março de 2006.


Prof. FRANCISCO DE ASSIS TENÓRIO DE CARVALHO
Coordenador da Pós-Graduação em Ciência da Computação do
Centro de Informática da Universidade Federal de Pernambuco.